PREFAȚĂ

Se consideră anul 1971, când INTEL a anunțat producerea primului microprocesor, ca momentul trecerii de la electronica implementată cablat la electronica realizată programat. Primul microprocesor 4004 a fost conceput de M. E. Hoff ca un procesor puțin simplificat pentru a putea fi implementat, la vremea aceea, pe un singur chip în siliciu. Inițial, microprocesorul nu a fost un *computer-on-a-chip*, deşi în timp a ajuns la acest stadiu; conceptul de microprocesor s-a dezvoltat şi spre alte implementări specifice: microcontroller, microcalculator, procesor de semnale (DSP). Oricare din aceste circuite, pentru a deveni parte a unui sistem, impune un proces de proiectare care se bazează în egală măsură atât pe componenta hardware cât şi pe componenta software.

Microprocesorul, sau celelalte variante ale sale, integrat într-un sistem aduc acestuia acea performanță, care atunci când este realizată de ființa umană este referită ca inteligență. În general, microprocesorul este integrat ca unitate centrală de procesare în sistemele digitale de calcul. Actual, aproape toate sistemele au o arhitectură de sistem digital particularizându-se în funcție de aplicare/utilizare. Aplicațiile care nu sunt, în sensul clasic, sisteme digitale de calcul sunt în general implementate pe bază de microprocesor sau DSP. Implementările sistemelor pe bază de microcontroller ori DSP devin simple şi ieftine –pentru că utilizează deja o componentă de serie, devin performante – pentru că utilizează o componentă care poate asigura suport pentru inteligență, devin flexibile – pentru că utilizează o componentă ce poate fi programată.

Forța care a impus sistemele pe bază de microcontroller constă în programabilitatea unei componente de serie. Pentru sistemele la care o componentă de serie, chiar programabilă în software, nu duce la rezultatele cele mai bune au apărut implementările pe bază de circuite ASIC (*Application Specific Integrated Circuits*). Spre deosebire de sistemele pe bază de microcontroller, unde funcționarea dorită se obține, în software, printr-un anumit program, la sistemele pe bază de circuite ASIC funcționarea dorită se obține prin structurarea hardware-ului cu un anumit program. Alegerea între o implementare pe bază de microcontroller, o implementare pe bază de circuit ASIC sau una combinată depinde de aplicația respectivă şi de cerințele impuse.

Prezenta carte constituie un material, dar fără a absolutiza, pentru abordarea sistemelor pe bază de microcontroller. Materialul prezentat şi CD-ul

alăturat asigură un traseu de la descrierea funcţionării unui microcontroller generic, dar şi cu exemplificări comerciale, până la analiza, sinteza implementarea şi testarea unui sistem. Deoarece oferta pe piață pentru microcontrollere este vastă şi variată, pe acest traseu s-a accentuat prezentarea comparativă şi critică încercând să se imprime cititorului analiza lucidă în fața avalanşei agresive a ofertelor de piață. Cartea a fost elaborată în cadrul programului Tempus RESUME (REtraining SUpport for small and Medium Enterprises) de la Universitatea TRANSILVANIA din Braşov, program care a avut printre obiective şi sprijinul dezvoltării de sisteme pe bază de microcontrollere în/şi pentru întreprinderi mici şi mijlocii. Ținând cont de această adresă cartea nu este destinată numai inginerilor în specialitatea de electronică şi ingineria calculatoarelor ci şi inginerilor de alte specialități care au o practică sau doresc să abordeze acest domeniu al sistemelor pe bază de microcontroller. Nu există actual produse de HiTech care să nu se bazeze direct sau indirect pe microprocesor sau microcontroller. În consecință, într-o economie de tranziție, cum este a noastră, care doreşte să se impună, formarea de personal care să ştie să proiecteze/implementeze/exploateze sisteme pe bază de microcontroller este o necesitate prioritară.

Mulțumim tuturor celor care ne-au sprijinit, colegilor Catedrei de Electronică şi Calculatoare, d-l asist. ing. Răzvan Brătucu şi mai ales domnilor profesori Marcian Cîrstea de la DMU Leicester - Marea Britanie, Josef Hoffmann de la Fachhochschule Karlsruhe - Germania, Dr. Peter Corcoran - National University of Ireland, MSC Eng.Petronel Bigioi, doctorand la Universitatea Transilvania Braşov precum şi studenților aceleiaşi universități.

Braşov, Ianuarie 2001

CUPRINS

DESCRIERE GENERALĂ 1

1.1 ISTORIC 1
1.2 SCHEMA BLOC GENERALĂ 3
1.3 ARHITECTURA MC 5
1.3.1 Unitatea centrală de prelucrare 6
1.3.2 Memoria 8
1.3.3 Dispozitive I/O 10
1.4 SISTEMUL DE ÎNTRERUPERI 28
1.5 MANAGEMENTUL PUTERII 34
1.6 SCHEMA BLOC A UNUI MC 35
1.7 FAMILII DE MC 37
1.8 CLASIFICAREA MC 41

PROGRAMAREA SISTEMELOR CU MC 43

2.1 PROIECTAREA PROGRAMELOR DE APLICAȚIE 43
2.1.1 Generalități 43
2.1.2 Instrucțiuni ale MC 50
2.1.3 Instrumente software de proiectare: MC 8051 56
2.1.4 Instrumente software de proiectare: MC PIC 67

MICROCONTROLLERE MOTOROLA 71

3.1 MC M68HC05 72
3.1.1 Memoria internă 72
3.1.2 Unitatea centrală 73
3.1.3 Moduri de adresare 75
3.1.4 Setul de instrucțiuni 76
3.1.5 Sistemul de întreruperi 77
3.1.6 Interfețe şi periferice on-chip 78
3.1.7 Managementul puterii 87
3.1.8 Autoverificarea 88
3.1.9 Programarea EPROM 88
3.2 MC M68HC08 89
3.2.1 Unitatea centrală 6808 90
3.2.2 Interfețe şi periferice on-chip 91
3.2.3 Programarea memoriei EEPROM 96
3.3 MC PE 16 BIȚI - 6816 97
3.3.1 Modulul de integrare (SIM, System Integration Module) 98
3.3.2 Interfețe 98
3.4 MC PE 32 DE BIȚI - 68300 99
3.5 DATE COMPARATIVE PENTRU MC MOTOROLA –CISC 100

FAMILIA MCS-51 101
4.1 STRUCTURA ŞI FUNCȚIONAREA 102
4.1.1 Gestionarea memoriei 104
4.1.2 Circuitele timer 104
4.1.3 Interfața serială UART 106
4.1.4 Sistemul de întreruperi 106
4.1.5 Operarea cu economie de energie 108
4.2 PROGRAMAREA MC DIN FAMILIA MCS-51 108
4.2.1 Setul de instrucțiuni 8051 108
4.2.2 Modurile de adresare 110
4.3 ALTE INTERFEȚE ŞI PERIFERICE ON-CHIP 111
4.3.1 Interfața I^2C (Siemens P80CL580) 111
4.3.2 Aria de numărătoare programabilă (PCA) 113
4.3.3 MC cu interfață pentru RAM nevolatil - NVRAM (Dallas DS5000FP) 115
4.3.4 MC specializat pentru TV şi video (PHILIPS 83C145) 115
4.3.5 MC cu arie configurabilă (TRISCEND E5) 116
4.4 SISTEM MINIMAL CU 8051 118
4.5 DATE COMPARATIVE PENTRU MC DIN FAMILIA MCS-51 119
MICROCONTROLLERE RISC 121
5.1 MICROCONTROLLER PIC 121
5.1.1 PIC12 121
5.1.2 PIC16 125
5.1.3 PIC17 127
5.2 MC ATMEL 129
5.2.1 Familia AVR 129
5.2.2 Familia ARM 138
CRITERII DE PROIECTARE 141
6.1 CRITERIILE PENTRU ALEGEREA UNUI MC 141
6.2 ALGORITMUL PROIECTĂRII SISTEMELOR CU MC 143
6.3 PROIECTAREA SISTEMELOR CU MC ÎN VEDEREA SIGURANȚEI ÎN EXPLOATARE 145
6.3.1 Cablajul imprimat 146
6.3.2 Ceasul de gardă 147
6.3.3 Programarea defensivă 147
APLICAȚII 149
7.1 IMPLEMENTAREA UNEI APLICAȚII SIMPLE DE COMANDĂ ŞI CONTROL DIGITAL 149
7.1.1 Varianta de implementare cu microcontroller CISC 151
7.1.2 Varianta de implementare cu microcontroller RISC. 163
7.2 IMPLEMENTAREA INTERFEȚELOR LA PROCES 181
7.2.1 Convertoare A/D controlabile serial 182
7.2.2 Interfațarea paralelă a convertoarelor A/D 189
7.2.3 Convertoare A/D complexe 195
7.2.4 Convertor D/A controlabil serial 201
7.2.5 Programarea unui sistem de dozare gravimetrică dotat cu MC 205
BIBLIOGRAFIE 219

1

DESCRIERE GENERALĂ

1.1 ISTORIC

Privind evoluția istorică a operației de comandă a unui proces putem contura imaginea unui microcontroller (MC - se va folosi în continuare această prescurtare pentru a numi un microcontroller). Un controller este un sistem folosit pentru a comanda şi a prelua stări de la un proces sau un aspect al mediului înconjurător. La început un controller era un echipament de mari dimensiuni. După apariția microprocesoarelor dimensiunile controller-elor s-au redus. Procesul de miniaturizare a continuat, toate componentele necesare unui controller au fost integrate pe acelaşi chip. S-a născut astfel calculatorul pe un singur chip specializat pentru implementarea operațiilor de control; acesta este microcontroller-ul. Un microcontroller este un circuit realizat pe un singur chip care conține tipic:

- unitatea centrală;
- generatorul de tact (la care trebuie adăugat din exterior un cristal de cuarț sau în aplicații mai puțin pretențioase, un circuit RC);
- memoria volatilă (RAM);
- memoria nevolatilă (ROM/PROM/EPROM/EEPROM);
- dispozitive I/O seriale şi paralele;
- controller de întreruperi, controller DMA, numărătoare/temporizatoare (*timers*), covertoare A/D şi D/A, etc.;
- periferice.

Prețul unui MC este redus din cauza cantităților mari în care se fabrică. Prețul mic al MC aduce cu sine şi micşorarea prețului sistemelor de control (se micşorează inclusiv costul proiectării).

Diferențele dintre microprocesor şi unitatea centrală a MC se atenuează în timp. Astfel, marii constructori de procesoare au realizat nişte circuite care s-ar putea numi super-microcontroller-e, aşa cum sunt Motorola 68EC300, INTEL 386EX sau IBM PowerPC 403GB, care sunt microcalculatoare pe un singur chip.

Cu un MC se poate realiza un controller integrat (*Embedded Controller, EC*). Un controller integrat face parte dintr-un sistem construit cu un anumit scop, altul decât calcule generale. Pe lângă MC, un controller integrat are nevoie de hardware suplimentar pentru a-şi îndeplini funcţia.

Importanţa MC este dovedită incontestabil de piaţa care este în continuă creştere. Astfel, evoluţia vânzărilor de MC în lume se reflrctă în tabelele 1.1 şi 1.2.

Tabelul 1.1

Evoluţia vânzărilor MC în lume (în milioane dolari)

MC	'90	'92	'94	'96	'97	'98	'99 (estimat)	'00 (estimat)
4-bit	1393	1596	1761	1849	1881	1856	1816	1757
8-bit	2077	2862	4689	6553	7529	8423	9219	9715
16-bit	192	340	810	1628	2191	2969	3678	4405

Tabelul 1.2

Evoluţia vânzărilor MC în lume (în milioane bucăţi)

MC	'90	'92	'94	'96	'97	'98	'99 (estimat)	'00 (estimat)
4-bit	778	979	1063	1100	1096	1064	1025	970
8-bit	588	843	1449	2123	2374	2556	2681	2700
16-bit	22	45	106	227	313	419	501	585

Principalii producători de microprocesoare au realizat în 1994 şi 1995 următoarele cifre de afaceri (tabelul 1.3).

Tabelul 1.3

Cifre de afaceri realizate în anii 1994, 1995 din vânzarea de microprocesoare (în milioane dolari)

Producător	1994	1995
Intel	10800	8036
AMD	881	992
Motorola	781	827
IBM	468	297
TI	219	202
Cyrix	210	240
Hitachi	188	66
NEC	100	82
LSI Logic	58	51
IDT	45	25

În prezent datele arată modificări de structură; de exemplu cifra de afaceri a AMD se apropie de cea a lui INTEL. Cifra de afaceri, ca ordin de mărime, era deja în anii 1994, 1995 apropiată de cea realizată din vânzări de microprocesoare (tabelul 1.4).

Tabelul 1.4

Cifre de afaceri realizate în anii 1994, 1995 din vânzarea de microcontrollere (în milioane dolari)

Producător	1994	1995
Motorola	1781	1511
NEC	1421	1208
Mitsubishi	945	708
Hitachi	899	782
Intel	835	605
TI	807	534
Philips	524	345
Matsushita	500	413
Lucent (AT&T)	492	275
Toshiba	400	328

1.2 SCHEMA BLOC GENERALĂ

Vom defini un microcontroller pornind de la o reprezentare simplificată a sa în interacțiune cu mediul (figura 1.1).

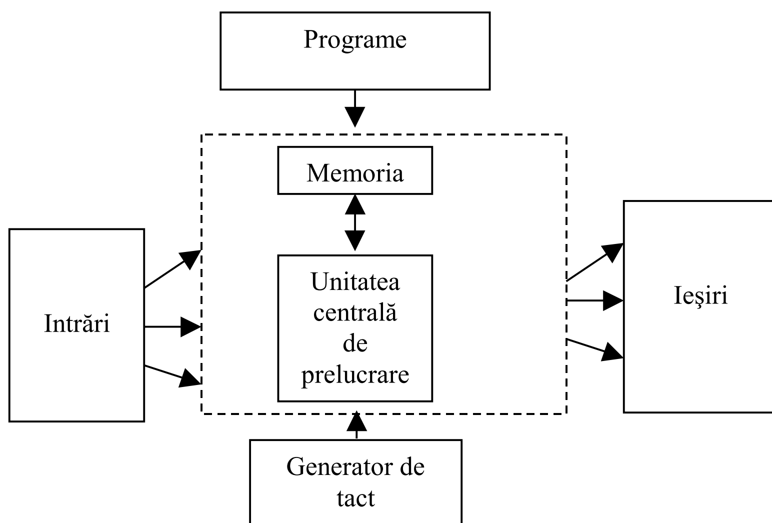


Figura 1.1 Schema simplificată a unui microcontoller

Ca intrări se folosesc de regulă semnale provenind de la comutatoarele individuale sau de la traductoare (de temperatură, de presiune, foto, traductoare specializate). Intrările pot fi digitale sau analogice. Intrările digitale vehiculează semnale discrete, informaţia "citită" fiind informaţia ce se eşantionează la momentul citirii liniei respective. Intrările analogice vehiculează informaţii exprimabile prin funcţii continue de timp. "Citirea" acestora de către microcontroller presupune prezenţa unor circuite capabile să prelucreze aceste informaţii, fie comparatoare analogice, fie convertoare analog-numerice, ale căror ieşiri sunt citite de către MC.

Ieşirile sunt fie analogice, caz în care reprezintă de fapt ieşiri ale convertoarelor numeric-analogice, fie sunt digitale, caz în care informaţia este în general memorată pe acestea până la o nouă scriere operată de către UC la un port al MC. Ieşirile pot comanda dispozitive de afişare, relee, motoare, difuzoare, etc.

O structură mai detaliată, care include blocurile principale, este reprezentată în figura 1.2.

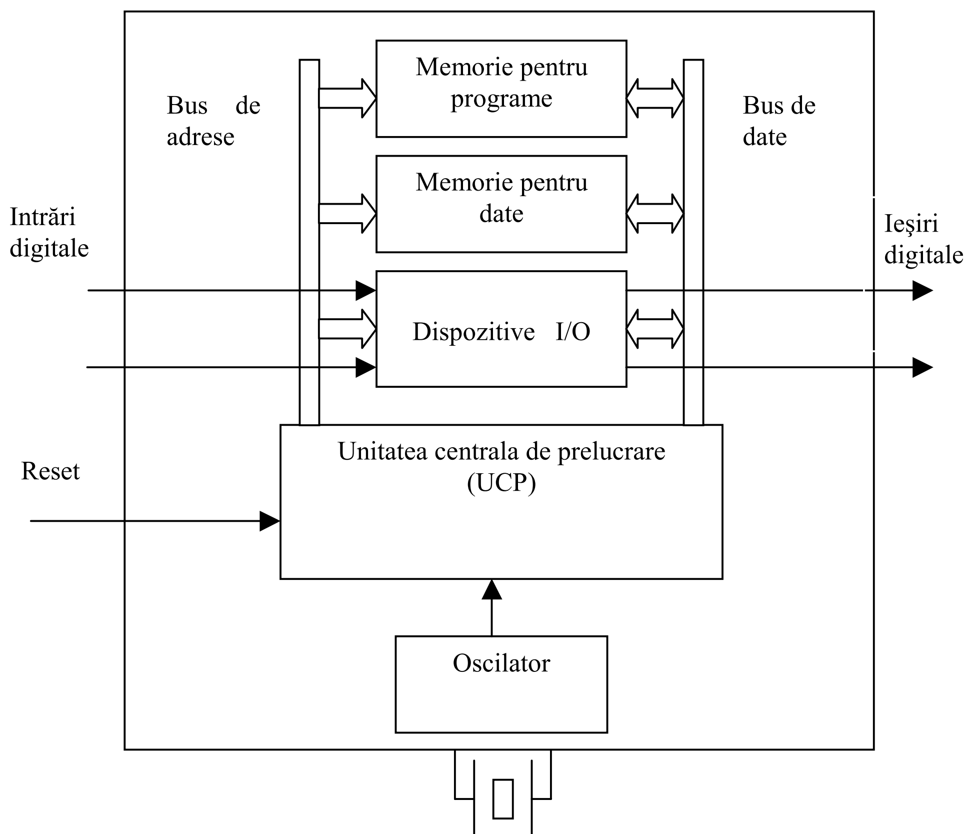


Figura 1.2 Schema bloc a unui microcontroller

Se poate distinge o primă diferență față de calculatoarele personale, unde intrările se fac de regulă de la tastatură şi ieşirile pe monitorul TV sau la imprimantă. Dacă un calculator personal este folosit pentru a prelucra informații şi a afişa rezultatele pe monitor sau hârtie, un MC comandă un proces.

Un element important, fără de care un MC nu poate funcționa, este programul (sau programele), care se stochează în memoria proprie MC.

Un MC poate fi definit ca un sistem de calcul complet pe un singur chip. Acesta include o unitate centrală, memorie, oscilator pentru tact şi dispozitive I/O. Un MC poate fi privit ca un microprocesor care pe acelaşi chip mai conține memorie şi o serie de interfețe. Natura şi complexitatea aplicației în care este folosit MC determină performanțele unității centrale, capacitatea de memorie şi tipul interfețelor ce compun structura internă a MC.

1.3 ARHITECTURA MC

Arhitectura unui MC defineşte atributele sistemului aşa cum sunt ele văzute de un programator în limbaj de asamblare. Deoarece un microcontroller este un caz particular de calculator, (calculator specializat pentru operații I/O, realizat pe un singur chip), acesta este compus din cele cinci elemente de bază: unitate de intrare, unitate de memorie, unitate aritmetică şi logică, unitate de control şi unitate de ieşire. Unitatea de control împreună cu unitatea aritmetică şi logică compun împreună unitatea centrală de prelucrare pe care o vom referi în continuare prescurtat cu UCP. Unitățile de intrare şi ieşire vor fi tratate împreună şi vor fi referite prescurtat ca sistem I/O.

Blocurile componente ale MC sunt legate între ele printr-o magistrală internă (bus). Magistrala vehiculează semnale de adresă, de date şi semnale de control. Mărimea acestor magistrale constituie una dintre caracteristicile cele mai importante ale unui MC. Prin magistrala de adrese unitatea centrală de prelucrare (UCP) selectează o locație de memorie sau un dispozitiv I/O, iar pe magistrala de date se face schimbul de informație între UCP şi memorie sau dispozitivele I/O. Între UCP şi memorie sunt transferate atât date cât şi instrucțiuni. Acestea se pot transfera pe o singură magistrală de date sau pe magistrale de date diferite.

Arhitectura von Neumann prevede existența unui bus unic folosit pentru circulația datelor şi a instrucțiunilor. Când un controller cu o astfel de arhitectură adresează memoria, bus-ul de date este folosit pentru a transfera întîi codul instrucțiunii, apoi pentru a transfera date. Accesul fiind realizat în 2 paşi, este destul de lent.

Arhitectura Harvard prevede un bus separat pentru date şi instrucțiuni. Când codul instrucțiunii se află pe bus-ul de instrucțiuni, pe bus-ul de date se află

datele instrucțiunii anterioare. Structura MC este mai complexă, dar performanțele de viteză sunt mai bune.

Magistralele interne, după numele lor, nu sunt accesibile în exterior la nivel de pin. Această stare caracterizează regimul normal de funcționare. În regimuri speciale de funcționare, numite diferit la diferitele tipuri de MC, semnalele magistralelor de adrese şi de date sunt accesibile la nivel de pin sacrificând funcțiile inițiale ale pin-ilor respectivi. Totodată este necesar să fie livrate în exterior şi semnale de comandă (cel puțin comandă de scriere – WR şi comandă de citire – RD). Această facilitate este utilă pentru extensii de memorie sau pentru a ataşa sistemului interfețe suplimentare. Ea nu este posibilă în mod normal de funcționare deoarece aceasta ar presupune existența unui număr foarte mare de pini, ceea ce ar fi neeconomic. Magistralele de adrese şi de date pot fi aduse la pin multiplexat sau nemultiplexat. Familia MCS-51 este un exemplu pentru acces multiplexat la magistrale. Accesul se face în doi paşi; în primul pas se conectează liniile de adresă la pinii portului "sacrificat", iar în al doilea pas se conectează datele la aceiaşi pini. Pentru a putea utiliza informația de adresare, adresa se memorează într-un latch, de asemenea exterior microcontroller-ului. Complicarea accesului la magistralele interne prin multiplexare este justificată de asemenea din rațiunea menținerii unui număr cât mai mic de pini la capsulă. MC mai puțin performante (ex. - MC low cost) nu au magistralele interne accesibile la nivel de pin.

Ca urmare a celor prezentate se poate aprecia performanța unui MC din punct de vedere al magistralei interne după următoarele criterii:

- dimensiunea magistralei de date;
- dimensiunea magistralei de adrese;
- accesibilitatea în exterior la magistrale;
- magistrale accesibile multiplexat sau nemultiplexat.

1.3.1 Unitatea centrală de prelucrare

Unitatea centrală de prelucrare este compusă din unitatea aritmetică şi logică (UAL) şi din unitatea de control.

Unitatea aritmetică şi logică este secțiunea responsabilă cu efectuarea operațiilor aritmetice şi logice asupra operanzilor ce îi sunt furnizați. Modul de implementare a operațiilor este transparent pentru utilizator; important pentru utilizatorul de MC este repertoriul operațiilor implementate pentru a aprecia posibilitatea implementării optime a aplicației particulare de control. De asemenea este important timpul de execuție al fiecărei operații pentru a aprecia dacă timpul necesar procesării complete satisface cerințele de timp ale aplicației.

Unitatea ce control este responsabilă cu decodificarea codului operației conținut de codul unei instrucțiuni. Pe baza decodificării unitatea de control

elaborează semnale pentru comanda celorlalte blocuri funcționale pentru a finaliza executarea unei instrucțiuni. Modul de implementare al acestui bloc este de asemenea transparent utilizatorului.

Unitatea centrală de prelucrare conține un set de registre interne, similare unor locații de memorie, folosite pentru memorarea unor date des apelate sau pentru programarea unor anumite funcții. Diferitele familii de MC folosesc seturi diferite de registre. Există însă câteva registre comune.

A (*Accumulator*) - registrul acumulator - este folosit deseori pentru a stoca un operand şi rezultatul operației aritmetice sau logice.

PC (*Program Counter*) – registru numărător de program - este registrul care stochează adresa următoarei instrucțiuni de executat. După un RESET (inițializarea MC), registrul PC se încarcă dintr-o locație de memorie numită vector de reset. Această locație conține adresa primei instrucțiuni de executat. PC este incrementat automat la execuția unei instrucțiuni.

SP (*Stack Pointer*) – registrul indicator de stivă - conținutul acestui registru indică adresa curentă a stivei. Stiva reprezintă o zonă de memorie accesibilă rapid în care se depun temporar informații importante în desfăşurarea programului. Stiva este definită de obicei în RAM. Implementarea accesului presupune existența unui registru de adresare (SP) şi a mecanismului de memorare declanşat de instrucțiuni specifice (instrucțiunile PUSH/POP).

Un aspect important ce se reflectă în arhitectura unui MC este modul de programare. Prin arhitectură înțelegem felul în care sunt dispuse "resursele" unui MC. Cu cât structura, funcționalitatea şi accesul la acestea sunt mai profund standardizate şi simetrizate, cu atât numărul de instrucțiuni de care dispune MC este mai redus şi viteza de execuție a programelor creşte.

Conceptul CISC (Complex Instruction Set Computer) pe baza căruia sunt construite majoritatea MC, prevede existența unui număr mare de instrucțiuni (tipic >80), ceea ce face mai uşoară munca programatorului. Multe din instrucțiuni sunt specializate, adică se pot folosi doar în anumite moduri de adresare sau cu anumite registre.

Evoluția MC este spre arhitectura RISC (Reduced Instruction Set Computer), în cadrul căreia un MC are un număr mic de instrucțiuni. Avantajele sunt un chip mai mic, cu un număr de pini mai mic, cu un consum mai redus şi cu o viteză mai mare. Instrucțiunile sunt simetrice, adică pot fi folosite la fel în orice mod de adresare sau cu orice registru, nu au excepții sau restricții.

În prezent un MC este cu mai mult decât o arhitectură RISC, el este cu o arhitectură SISC (Specific ISC). Instrucțiunile sunt specifice pentru a lucra optim cu dispozitivele I/O, permit manipularea la nivel de bit şi sunt mai puține instrucțiuni de uz general, aşa cum întâlnim la microprocesoarele 8086, 68000 etc.

În general MC cu arhitectură CISC dispun de un număr redus de registre. Acestea au funcții determinate (acumulator, registru index,etc), iar MC cu arhitectură RISC dispun de un număr mai mare de registre cu un mare grad de ortogonalitate ceea ce înseamnă că pot fi folosite simetric în orice mod de adresare (ATMEL AVR are 32 de registre)

Registrele interne pot avea diferite dimensiuni, astfel registrele acumulator au dimensiunea egală cu cea a magistralei de date (8, 16 sau 32 biți). Registrul numărător de program are dimensiunea egală cu cea a magistralei de adrese. Registrul indicator de stivă poate avea dimensiunea mai mică decât a magistralei de adrese permițând localizarea stivei într-o zonă restrânsă a memoriei RAM. Există şi MC cu stivă automată, acestea nu au nevoie de indicator de stivă dar au dezavantajul că nu se pot folosi mai mult de 2-4 nivele de stivă (MC din familia PIC).

1.3.2 Memoria

MC folosesc diferite tipuri de informații, care sunt stocate în diferite tipuri de memorii. Instrucțiunile care controlează funcționarea MC trebuie stocate într-o memorie nevolatilă, unde informațiile se păstrează şi după oprirea şi repornirea sursei de alimentare. Rezultatele intermediare şi variabilele pot fi înscrise într-o memorie volatilă, la acestea este important să se poată face scrierea /citirea rapid şi simplu în timpul funcționării.

Memoria RAM (*Random Access Memory*) este o memorie volatilă care poate fi citită sau scrisă de unitatea centrală. Locațiile din RAM sunt accesibile în orice ordine. Pe chip, memoria RAM ocupă mult loc şi implicit costurile de implementare sunt mari. De aceea un MC include de obicei puțin RAM. Memorie RAM static alimentată de la baterie se foloseşte pentru stocarea nevolatilă a cantităților mari de date, la o viteză de acces mare şi cu un număr nelimitat de ştergeri şi reînscrieri.

Memoria ROM (*Read Only Memory*) este cea mai ieftină şi cea mai simplă memorie şi se foloseşte la stocarea programelor în faza de fabricație. Unitatea centrală poate citi informațiile, dar nu le poate modifica.

Memoria PROM (*Programmable Read Only Memory*) este similară cu memoria ROM, dar ea poate fi programată de către utilizator. După posibilitățile de ştergere, această memorie poate fi de mai multe feluri:

Memoria EPROM (*Erasable PROM*) care se poate şterge prin expunere la raze ultraviolete. MC cu EPROM au un mic geam de cuarț care permite ca *chip-ul* să fie expus la radiația ultravioletă. Ştergerea este neselectivă, adică se poate şterge doar întreaga informație şi nu numai fragmente. Memoria poate fi ştearsă şi reînscrisă de un număr finit de ori. Programarea EPROM-ului necesită o procedură specială, iar MC cu EPROM au nevoie de regulă pentru înscrierea EPROM-ului de o tensiune auxiliară, de 12 V de exemplu. Unele MC au incluse circuite de programare a memoriei EPROM, cu ajutorul cărora unitatea centrală poate programa memoria EPROM. În timpul programării memoria EPROM nu este conectată la magistrala de date şi adrese. Unele MC sunt prevăzute cu mod special de lucru, în care sunt văzute din exterior ca nişte memorii EPROM obişnuite şi pot fi astfel programate cu orice programator.

OTP (*One Time Programmable PROM*) se foloseşte pentru multe serii de MC. Memoria OTP este de fapt o memorie EPROM, dar *chip-ul* a fost capsulat într-o capsulă de material plastic fără fereastră, care este mult mai ieftină. Memoria nu se poate şterge sau reprograma. Preţul unui MC cu OTP este mic, viteza este bună, dar aplicaţiile sunt lipsite de flexibilitate.

Memoria EEPROM (*Electrically Erasable Programmable Read Only Memory*) poate fi ştearsă electric de către unitatea centrală, în timpul funcţionării. Ştergerea este selectivă, iar pentru reînscriere trebuie parcurşi mai mulţi paşi. Memoria EEPROM echipează multe MC, fiind ieftină. În memoria EEPROM se memorează un mic număr de parametri care se schimbă din timp în timp. Memoria este lentă şi numărul de ştergeri/scrieri este limitat (tipic 10 000).

Memoria FLASH este o memorie asemănătoare EPROM şi EEPROM în sensul că poate fi ştearsă şi reprogramată în sistemul în care este folosită (fără a fi necesar un sistem dedicat). Are capacitatea unui EPROM, dar nu necesită fereastră pentru ştergere. Ca şi EEPROM, memoria FLASH poate fi ştearsă şi programată electric. Memoria FLASH nu permite ştergerea individuală de locaţii, utilizatorul poate să şteargă doar întregul conţinut.

Din punct de vedere al locului şi modului de programare a memoriilor de tip PROM există două concepte:

- ICP (*In Circuit Programming*) – programarea memoriei când MC se află pe placa de cablaj imprimat a aplicaţiei;
- ISP (*In System Programming*) – presupune posibilitatea de reprogramare în funcţionare a sistemului. De exemplu la autovehiculele comandate de MC, schimbarea tipului benzinei sau schimbarea unei legi privitoare la poluare pot fi actualizate în programul MC fără ca acesta să fie scos din autovehicul.

În funcţie de numărul aplicaţiilor în care urmează să fie folosit MC se recomandă folosirea MC cu ROM pentru volum mare de producţie (ROM se înscrie cu mască la fabricant), OTP pentru volum mic de producţie şi EPROM pentru prototipuri.

Tipul de memorie şi capacitatea memoriei din echiparea unui MC sunt caracteristici particulare fiecărui tip de circuit. Printre alte diferenţe, aceşti parametri sunt diferiţi pentru membrii unei aceleiaşi familii de MC. Se vor prezenta date concrete pentru exemplificare într-un tabel recapitulativ la sfârşitul capitolului (tabelul 1.6).

Pentru a aprecia un MC sub aspectul componentei memorie este necesar să se considere următoarele caracteristici:

- varietatea tipurilor de memorie pe chip: RAM, ROM /OTP /EPROM /EEPROM /FLASH;
- capacitatea memoriei aflată pe chip;
- uşurinţa cu care se poate programa (programare în circuit sau nu, necesitatea unor tensiuni de programare suplimentare).

1.3.3 Dispozitive I/O

Dispozitivele I/O reprezintă un aspect de mare interes atunci când este vorba de microcontroller-e, interes rezultat din însăşi particularitatea unui MC: aceea de a interacționa cu mediul în procesul de control pe care îl conduce.

Dispozitivele I/O implementează funcții speciale degrevând unitatea centrală de toate aspectele specifice de comandă şi control în funcția respectivă. Există o varietate mare de dispozitive I/O; dispozitivele I/O conduc operații generale de comunicație (transfer serial sau paralel de date), funcții generale de timp (numărare de evenimente, generare de impulsuri), operații de conversie analog/numerică, funcții de protecție, funcții speciale de comandă, şi enumerarea poate continua. Parte din resurse acoperă funcțiile de control propriu-zis, iar o parte asigură funcțiile necesare aplicațiilor în timp real (sistemul de întreruperi, timer).

Din această mare varietate, parte din dispozitive se găsesc în configurația tuturor MC sau sunt foarte des întâlnite, iar o altă parte de dispozitive o regăsim doar în MC construite pentru a optimiza aplicații cu un grad mare de particularitate. În acest capitol, în continuare, vor fi prezentate dispozitive des întâlnite în echiparea MC. În capitolele următoare, pe măsură ce vor fi prezentate exemple de MC, vor fi descrise şi o serie de dispozitive I/O speciale ce intră în componența acestora.

Dispozitivele I/O sunt "văzute" de unitatea centrală ca porturi. Există două filozofii de alocare a adreselor (mapare) pentru apelarea porturilor de către unitatea centrală. Porturile sunt mapate fie în spațiul de memorie, fie într-un spațiu propriu. Cele două metode sunt exemplificate la MC produse de Motorola şi MC produse de Intel. La MC Motorola dispozitivele I/O ocupă adrese într-un spațiu comun cu memoria, ceea ce atrage după sine reducerea numărului de locații de memorie. MC de la Intel folosesc un semnal care indică dacă la adresa curentă se apelează o locație de memorie sau un dispozitiv I/O.

Avantajele mapării în spații separate:

- Metoda nu este susceptibilă de a provoca erori soft deoarece se folosesc instrucțiuni diferite pentru a accesa memoria şi dispozitivele I/O;
- Dispozitivele I/O nu ocupă spațiu de memorie;
- Decodificatorul de adrese pentru dispozitivele I/O este mai simplu deoarece spațiul de adresare al dispozitivelor I/O este mai mic.

Dezavantaje ale mapării în spații separate:

- instrucțiunile I/O sunt mai puțin flexibile în comparație cu instrucțiunile de lucru cu memoria;
- instrucțiunile I/O nu permit folosirea modurilor de adresare folosite în lucrul cu memoria.

Avantajul mapării în spațiu comun:

- toate instrucțiunile de lucru cu memoria şi toate modurile de adresare sunt utilizabile şi în tratarea dispozitivelo I/O (programarea operațiilor I/O este foarte flexibilă);

Dezavantajele mapării în acelaşi spațiu:

- este susceptibil la erori software;
- spațiul de adresare disponibil pentru memorie este mai mic.

Avantajul mapării într-un spațiu comun este acela că se poate opera cu porturile la fel ca şi cu o locație de memorie; multe operații aritmetice şi logice folosesc operanzi direct de la port, datele pot fi mutate între porturi cu o singură instrucțiune, conținutul unui port poate fi citit, incrementat şi rezultatul scris din nou la port de asemenea cu o singură instrucțiune.

Se conturează trei direcții de evoluție a MC din punct de vedere al resurselor disponibile pe chip. O direcție este dezvoltarea de MC de uz general care sunt puse la dispoziția utilizatorului pentru a realiza aplicații "de la A la Z". O a doua direcție o reprezintă dezvoltarea de MC specializate care dispun de resurse specifice unui anume gen de aplicații (automobile, telecomunicații, Internet, acționări electrice, etc). Pentru aceste MC specializate atât structura hardware cât şi instrucțiunile sunt specifice şi sunt calate pe un gen particular de aplicații. Există o a treia tendință de dezvoltare de MC care dispun de resurse hardware configurabile la utilizator. Între resursele configurabile se află memoria, elemente de logică programată, module specializate, gen convertoare A/D şi convertoare D/A, şi module de procesare avansată a datelor.

Porturi paralele

Porturile paralele I/O facilitează transferul simultan al mai multor biți între MC şi mediu. Numărul de biți transferați ca urmare a executării unei instrucțiuni depinde de organizarea particulară a porturilor în cadrul MC; unele porturi sunt organizate pe 8 biți, altele pe un număr mai mare de biți. Sensul transferului, I (*input*) sau O (*output*) este programabil şi se stabileşte de obicei într-un registru de sens. Pentru a veni în întâmpinarea cerințelor specifice de interacțiune cu mediul sunt implementate mecanisme de apelare la nivel de bit pentru registrele porturilor paralele I/O. Mecanismul presupune o structură hardware corespunzătoare şi instrucțiuni pentru manipularea informației la nivel de bit (un operand este un bit). Astfel, la execuția unei instrucțiuni orientată pe bit, există posibilitatea de a transfera un singur bit, de a masca biți care nu sunt folosiți într-o operație particulară şi de a efectua operații logice pe un singur bit al unui port paralel I/O. Datorită capabilității de manipulare la nivel de bit la descrierea caracteristicilor sistemului paralel I/O se specifică numărul total de linii I/O.

MC68HC705C8 dispune de 31 de linii I/O digitale de uz general grupate în patru porturi (A, B, C şi D). Porturile A, B, şi C sunt porturi de câte opt biți fiecare şi pun

la dispoziție în total 24 de linii care pot fi folosite fie ca intrări, fie ca ieşiri. Portul D dispune de 7 linii care pot fi folosite doar ca intări.

Circuitul de interfață al portului paralel este responsabil de corectitudinea transferului cu dispozitivul I/O. În acest scop se foloseste una din următoarele metode de sincronizare:

- Transfer direct (*brute-force method*) – în cadrul acestei metode interfața are doar rolul de a transfera semnalele de la unitatea centrală spre dispozitivul I/O (dacă este o operație de ieşire) sau de la dispozitivul I/O la unitatea centrală (dacă este o operație de intrare). Se foloseşte acest tip de transfer dacă se citeşte nivelul unui potențial, dacă se comandă la ieşire un nivel logic (pentru comanda unor LED-uri, de exemplu).
- Transfer strobat (*strobe method*) – această metodă foloseşte semnale de strobare pentru a indica starea stabilă a semnalelor la porturile de intrare sau de ieşire. Într-o operație de intrare, dispozitivul de intrare generează semnalul de strobare când datele sunt stabile la pinii portului de intrare al interfeței. Stocarea informației în registrul de date al interfeței se face cu semnalul de strobe. Daca este vorba de o operație de ieşire, când semnalele de date sunt stabile, interfața elaborează semnal de strobare cu care dispozitivul I/O preia efectiv datele. Aceată metodă poate fi folosită dacă interfața şi dispozitivul I/O lucrează la viteze comparabile.
- Transfer cu protocol (*handshake method*) – există cazuri în care vitezele de lucru ale celor două părți implicate în transfer sunt mult diferite şi metodele mai sus amintite nu oferă cadrul potrivit pentru transferul datelor. Soluția este ca interfața şi dispozitivul I/O să poarte un dialog numit handshake. Sunt necesare două semnale de protocol; unul din ele este elaborat de circuitul de interfață iar celălalt de dispozitivul I/O. Tranzacțiile de protocol diferă pentru operații de intrare şi de ieşire.

În figura 1.3 este ilustrată organizarea portului A al unui circuit particular (MC68HC705C8).

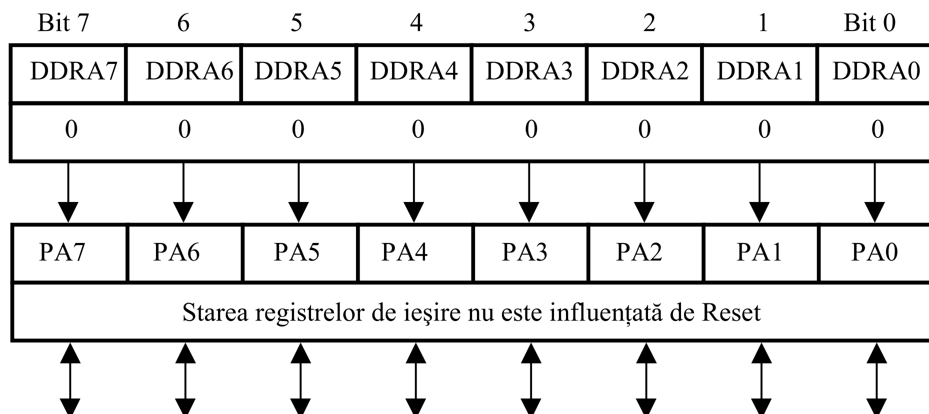


Figura 1. 3 Organizarea portului paralel A (MC68HC705C8)

Registrul DDRA (*Data Direction Register port A*) este un registru de 8 biţi ca şi registrul de date PA al portului A . Sub fiecare din registre este reprezentată starea iniţială a registrelor (după reset). În cazul particular considerat, portului PA este implicit port de intrare.

O structură posibilă de circuit pentru o linie bidirecţională de scriere/citire la porturi este redată în figura 1.4. Se poate constata modul în care acţionează bitul din registrul de sens asupra circuitului driver de ieşire.

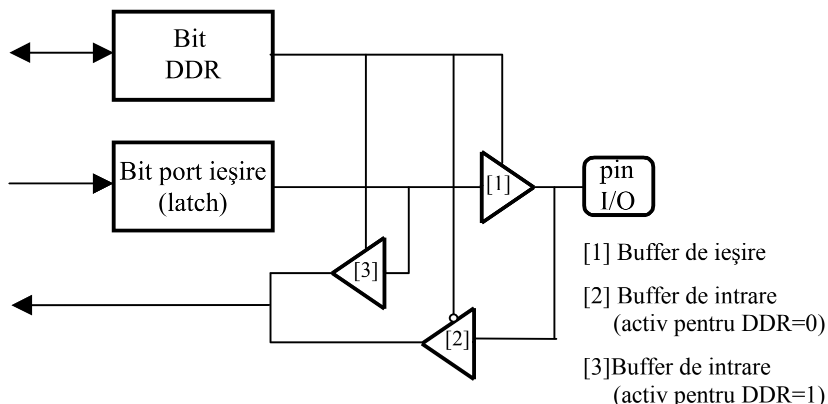


Figura 1.4 Circuit bidirecţional de scriere/citire la portul paralel

În partea stângă a diagramei sunt reprezentate legăturile spre magistrala internă de date a MC. În funcţie de natura operaţiei efectuate (citire sau scriere) şi de valoarea bitului din registrul de direcţie sunt posibile operaţiile exprimate în tabelul 1.5.

Tabelul 1.5

Funcţiile unui pin I/O

Operaţie	Bit DDR	Funcţie pin I/O
scriere	0	Pinul I/O este în mod intrare; datele pentru scriere vor fi înscrise în latch-ul de ieşire
scriere	1	Datele pentru scriere sunt înscrise în latch-ul de ieşire şi transferate de buffer-ul [1] pinului I/O
citire	0	Este citită starea pinului I/O
citire	1	Pinul I/O este în mod ieşire; este citită informaţia din latch-ul de ieşire.

Se poate observa că o linie de ieşire a unui port paralel este mai mult decât un latch simplu; totodată, informaţia înscrisă în latch-ul de ieşire nu se schimbă decât dacă se efectuează o nouă operaţie de scriere la port.

Unele porturi paralele sunt considerate cvasi-bidirecţionale, ceea ce înseamnă că o linie poate fi folosită în acelaşi timp ca ieşire şi ca intrare. Această flexibilitate este considerată de interes în aplicaţii şi este posibilă printr-un “truc”.

Figura 1.5 reprezintă structura unei linii cvasi-bidirecționale. Trucul constă în exploatarea facilităților unui etaj de ieşire cu drena în gol (*open-drain*) care în acest exemplu este pevăzut cu o rezistență conectată intern la sursa de alimentare.

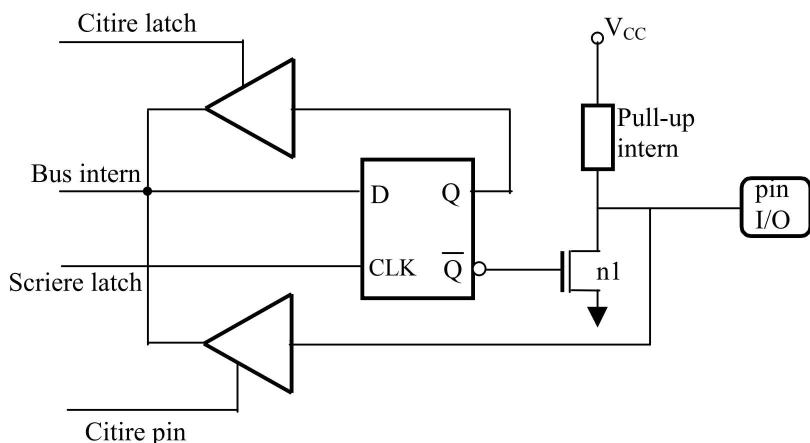


Figura 1.5 Structura unei linii cvasi-bidirecționale

Fiecare linie poate fi folosită independent ca intrare sau ca ieşire. Pentru a fi folosită ca intrare este necesar ca latch-ul să fie înscris cu unu logic (nQ va fi în acest caz zero) pentru ca tranzistorul n1 să fie blocat. Dacă nu este ataşat nimic la pin, rezistența internă de pull-up va determina citirea unui unu logic la pin. Dacă linia este forțată exterior în zero logic, etajul va genera curent prin rezistența de pull-up, iar informația citită va fi un zero logic. Există structuri mai complicate care în unele situații necesită conectarea în exterior a unei rezistențe de pull-up.

Pentru a interacționa cu mediul unui MC îi sunt necesare multe conexiuni la nivel de pin, ceea ce înseamnă un circuit cu un număr mare de pini. Din considerente de fiabilitate şi economice se caută reducerea numărului de pini ai unui circuit integrat. În această situație a fost necesar un compromis în urma căruia s-a ajuns la soluția prin care se atribuie unui pin funcții multiple (două sau mai multe). Pinii astfel sacrificați sunt pinii portului paralel. La pinii porturilor paralele sunt accesibile celelalte resurse ale circuitului; pot fi amintite conexiunile interfețelor seriale, conexiunile cu exteriorul ale modulelor timer, canalele de intrare ale convertoarelor A/D, liniile pentru generarea întreruperilor externe.

În mod normal de operare magistralele interne ale unui MC nu sunt accesibile în exterior, la nivel de pin. Există situații în care este necesară extensia resurselor (extensia memoriei, spre exemplu) caz în care este necesar accesul la magistralele interne. Pentru rezolvarea acestei probleme sunt folosiți tot pinii porturilor paralele. Din nou, pentru economie de pini “sacrificați“, aceiaşi pini sunt folosiți consecutiv pentru a vehicula adrese şi apoi date. Deoarece adresele folosesc liniile doar ca ieşiri, adresa este disponibilă la pinii portului paralel în timpul primului ciclu de ceas al execuției unei instrucțiuni. Adresa este memorată în circuite exterioare (în latch-uri). În al doilea ciclu de ceas se vehiculează pe aceste

linii datele. Modul de lucru în care magistralele interne sunt disponibile la pini este numit mod expandat de lucru.

Pentru a evalua capabilitatea porturilor paralele se recomandă considerarea următoarelor însuşiri:

- Numărul de linii I/O;
- Posibilitatea de a programa sensul liniilor I/O;
- Alte resurse accesibile prin funcții multiple la liniile I/O;
- Posibilitatea accesului la magistralele interne
- Aspecte electrice de interfațare.

Module de comunicații seriale

Comunicația serială este o metodă bine agreată şi în contextul MC deoarece oferă compatibilitate cu o gamă extinsă de dispozitive şi necesită un număr minim de fire, implicit un număr minim de conexiuni (pini puțini).

În transferul serial de date este necesar să se cunoască începutul şi sfârşitul informației transferate. Pentru a identifica cele două coordonate emițătorul şi receptorul trebuie să fie sincronizați. Sincronizarea se poate realiza prin trei metode, dintre care numai două sunt folosite în MC, urmând să le considerăm doar pe acestea în continuare. Oricare din metode presupune că durata unui bit este aceeaşi atât la emițător cât şi la receptor (este folosit acelaşi semnal de ceas pentru serializarea informației). Numărul de biți transmişi într-o secundă reprezintă rata de transfer numită baud rate; aceasta se măsoară în biți/secundă (bps). Durata unui bit este 1/(baud rate).

Modulul serial asincron

Prima metodă considerată este transferul serial asincron. Modulul serial asincron este referit ca UART (*Universal Asynchronous Receiver Transmiter*). Într-un transfer serial asincron începutul fiecărui byte este marcat de o tranziție a liniei menținută pe durata corespunzătoare unui bit. Sfârşitul este marcat de asemenea de un bit de stop; bitul de stop constă în menținerea liniei pe durata unui bit într-o stare predefinită. Între bitul de start şi bitul de stop sunt transferați biții de informație. O secvență completă este compusă dintr-un bit de start, opt sau nouă biți de date şi un bit de stop, în total 10 sau 11 unități de cod. Interfața serială asincronă poate fi implementată în două moduri într-un MC.

Cea mai puțin costisitoare soluție din punct de vedere al cheltuielilor de resurse hardware este un program care să genereze secvența de transfer ca şi componentă şi ca durată a fiecărui bit. Programul trebuie să măsoare durata fiecărui bit. Soluția este costisitoare din punct de vedere al timpului consumat în rularea programului.

A doua soluție de implementare a interfeței seriale este şi cea mai des întâlnită în MC. Interfața este realizat cu un modul hardware specializat. Unitatea centrală înscrie informația de transferat într-un registru al interfeței după care sarcina serializării şi a generării secvenței este finalizată de către hardware-ul interfeței. În cazul recepției interfața preia secvența recepționată, extrage elementele de sincronizare şi înscrie datele efective într-un registru. Datele recepționate sunt accesibile unității centrale în acest registru al interfeței. În cazul în care registrul de transmisie este gol sau registrul de recepție este plin interfața semnalizează starea şi poate chiar suspenda temporar execuția programului principal prin întreruperi hardware. O schemă bloc simplificată a unei interfețe seriale asincrone este reprezentată în figura 1.6.

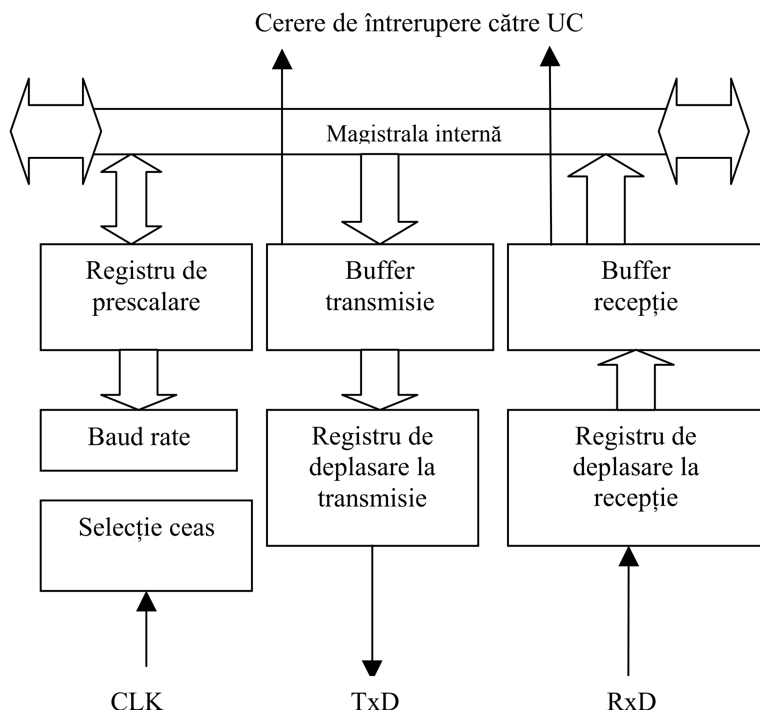


Figura 1.6 Schema bloc a UART

Ceasul pentru generarea ratei de transfer poate fi intern sau extern. În cazul în care se selectează ceas intern, acesta se formează din ceasul unității centrale cu o prescalare. Pentru ca rata de transfer să fie programabilă ceasul prescalat se divizează în continuare folosind un numărător al unui timer din resursele proprii. Interfața UART dispune de un registru de stare şi un registru de control cu ajutorul cărora se pot prgrama modul de lucru, parametrii de comunicație, selectarea ceasului, divizarea ceasului.

Conexiunile interfeței seriale asincrone sunt disponibile la pini cu funcții multiple la unul din porturile paralele.

Modulul serial sincron

Următoarele două metode de sincronizare posibile la transferul serial sunt transfer serial sincron şi transfer serial autosincronizat. Transferul autosincronizat se bazează pe utilizarea unui sistem special de codificare a datelor. Sistemul permite ca din datele recepţionate să se poată extrage semnalul de ceas. Această metodă nu este folosită în MC.

Transferul serial sincron este folosit pentru comunicaţii locale între procesoare sau cu dispozitivele periferice. În transferul sincron datele transferate sunt însoţite de semnalul de clock. Semnalul de clock este transferat pe o linie dedicată. Transferul serial sincron necesită trei conexiuni: emisie, recepţie şi clock. Interfaţa hardware este mai simplă decât la transferul asincron. Protocolul dintre emiţător şi receptor include o scurtă perioadă de sincronizare la iniţierea unui transfer de date. Spre deosebire de transferul asincron, la care sincronizarea se face prin bitul de start al fiecărui octet, la transferul sincron aceasta are loc o singură dată, la începutul transferului unui bloc de date. Transferul serial sincron permite rate de transfer mai mari decât transferul serial asincron; rate de 1Mbps.

Într-o conexiune serială sincronă unul din dispozitive este master iar celălalt este slave. Un transfer poate fi iniţiat doar de un master. Master-ul scrie un octet în registrul de transmisie de date al interfeţei seriale sincrone. Datele sunt transferate unui registru de deplasare unde sunt serializate şi transmise pe linia de emisie numită MOSI (*Master Output Slave Input*). Pe linia SCK (*Serial Clock*) este transmis clock-ul. La slave datele ajung tot pe linia MOSI într-un registru de deplasare unde sunt deplasate cu ceasul recepţionat pe linia SCK. La umplerea registrului de deplasare datele sunt transferate în registrul de recepţie de date.

Conexiunile interfeţei seriale sincrone sunt disponibile la pini cu funcţii multiple la unul din porturile paralele. Pinii asociaţi unei conexiuni seriale sincrone sunt următorii:

MISO – *Master In Slave Out* – Pinul MISO este configurat ca intrare într-un master şi ieşire dintr-un slave. MISO este conexiunea prin care se transmit date într-un sens (de la slave la master). Ieşirea MISO la un slave este în starea de înaltă impedanţă dacă dispozitivul slave nu este selectat.

MOSI – *Master Out Slave In* – Pinul MOSI este configurat ca ieşire dintr-un master şi intrare într-un slave. MOSI este cea de-a doua conexiune prin care se transmit date în celălalt sens (de la master la slave).

SCK – *Serial Clock* – Pinul SCK este ieşire la un master şi intrare la un slave. Prin această conexiune se transmite semnalul de sincronizare pentru transferul datelor pe liniile MISO şi MOSI. Pe durata a opt perioade ale SCK se transferă între master şi slave un byte de informaţie.

nSS – *non Slave Select* – La pinul nSS se aplică semnalul care selectează un dispozitiv slave; pentru un dispozitiv slave, semnalul trebuie să fie activ (low) pe toată durata unei tranzacţii în care este implicat. Pentru un dispozitiv master,

intrarea nSS se conecteaza permanent la 1 logic; în cazul în care la master, intrarea nSS devine 0, se seteaza un flag de eroare (MODF) în registrul de stare al interfeței seriale sincrone. Pinul nSS al unui master poate fi configurat şi ca ieşire de uz general prin setarea unui bit în registrul de sens DDRD. Ceilalți trei pini, amintiți anterior, sunt dedicați interfeței sincrone atâta timp cât este validată.

În sistemele care folosesc funcția serială sincronă există un master şi unul sau mai multe dispozitive slave. Există mai multe soluții de interconectare; acestea vor fi prezentate în continuare.

În cazul în care în sistem este un singur slave, conexiunea este făcută ca în figura 1.7.

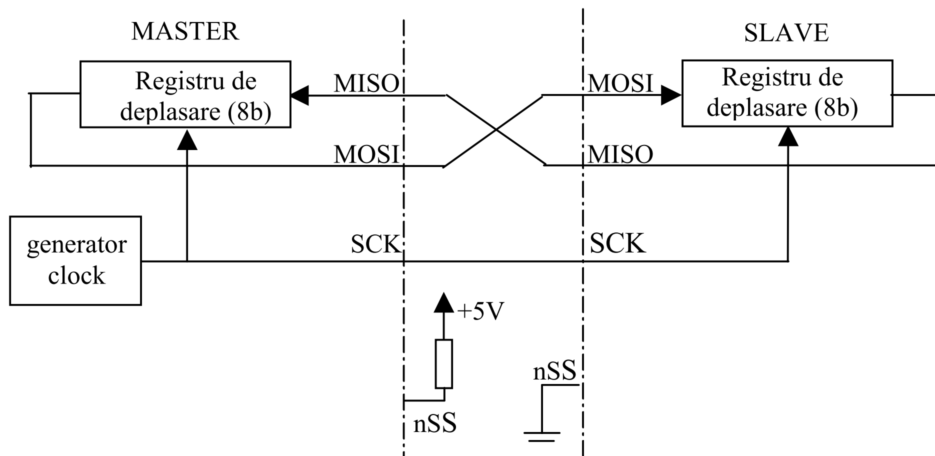


Figura 1.7 Conexiune serială sincronă master-slave

Interfața serială sincronă permite interconectarea mai multor dispozitive, dintre care unul singur este master. Există două soluții pentru o conexiune multi-slave. În una din ele master-ul foloseşte ieşiri de porturi pentru a selecta dispozitivul slave cu care se face transferul; în cealaltă soluție, toate dispozitivele slave sunt selectate şi sunt legate în inel. Conexiunea cu selecția fiecărui dispozitiv slave este reprezentată în figura 1.8.

Un alt mod de a realiza o conexiune serială sincronă în care există mai mult decât un slave este reprezentat în figura 1.9. Acest tip de conexiune diferă de precedentul prin următaorele:

- Pinul MISO al fiecărui slave este conectat la pinul MOSI al dispozitivului slave vecin. Pinul MOSI de la master este în continuare conectat la pinul MOSI de la primul slave.
- Pinul MISO de la master este conectat cu pinul MISO al ultimului dispozitiv slave.
- Intrările nSS ale tuturor dispozitivelor slave sunt conectate la masă. Nu este necesară comanadă de selecție de la master în această conexiune.

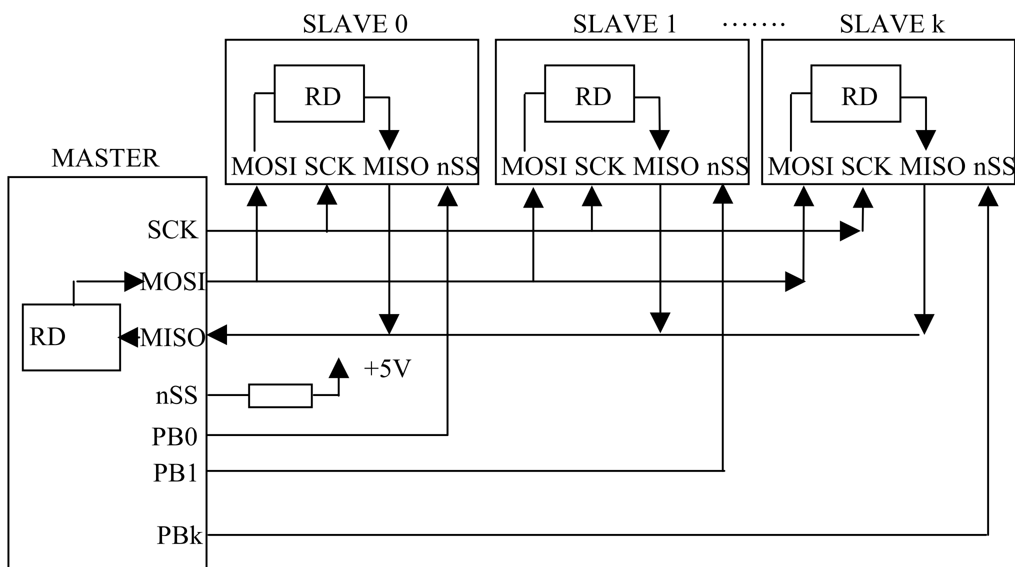


Figura 1.8 Conexiune serială sincronă single-master multi-slave (varianta 1)

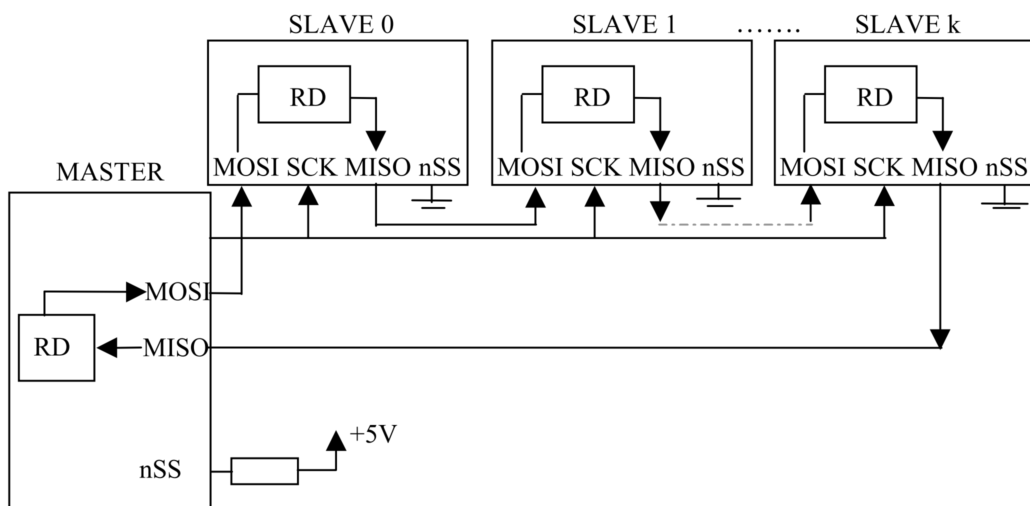


Figura 1.9 Conexiune serială sincronă single-master multi-slave (varianta 2)

Conexiunea din figura 1.9 foloseşte un număr minim de pini pentru a controla un număr mare de dispozitive. Ca dezavantaj, master-ul nu poate selecta un anume slave pentru transferul de date, datele trec prin toate dispozitivele din inel până la dispozitivul adresat.

Interfaţa serială sincronă este folosită pentru a comunica cu periferice cum ar fi un simplu registru de deplasare, un sistem de afişare LCD sau un sistem de conversie A/D. Modulul serial sincron este suficient de flexibil pentru a interfaţa

direct periferice cu standarde diferite, provenite de la diverşi producători. Conexiunea serială sincronă poate fi folosită şi pentru a extinde numărul de intrări/ieşiri acolo unde acesta este limitat de numărul de pini ai capsulei MC.

Interfețele seriale amintite sunt întâlnite în MC cu următoarele nume:

SCI (*Serial Communications Interface*) este un subsistem I/O serial indepenent, de tipul full duplex asincron (UART) numit astfel în MC Motorola.

SPI (*Serial Peripheral Interface*) este un modul serial folosit pentru a comunica sincron la distanțe mici cu viteze de până la 4 Mbps

SCI+ este similar cu SCI, are în plus suport pentru comunicații seriale sincrone. Dispune de o ieşire de ceas folosită pentru a transfera date în mod sincron cu un periferic de tip SPI.

SIOP (*Simple I/O Port*) este o implemenatre mai simplă a SPI.

Module Timer

Natura aplicațiilor pentru care s-a născut microcontroller-ul implică o multitudine de funcții de timp puse la dispoziția utilizatorului prin module de timp numite timer. Un MC este echipat în mod obligatoriu cu un astfel de modul mai mult sau mai puțin complex. Un sistem timer obişnuit pune la dispoziție un set de funcții implementate pe baza unui numărător liber central şi a unor blocuri funcționale pentru fiecare funcție în parte. Timer-ul are în structura sa, dintre toate celelalte subsisteme, cele mai multe registre. Toate funcțiile unui timer pot genera întreruperi independente; fiecare are controlul propriu şi propriul vector de întrerupere.

Modulele timer complexe sunt construite cu arii de numărare programabile (PCA). Pentru aplicații speciale în timp real s-au construit module timer cu unitate aritmetică şi logică proprie.

Timer-ul este folosit pentru a măsura timpul şi pentru a genera semnale cu perioade şi frecvențe dorite. Timer-ele nu sunt doar circuite cu funcții de temporizare; în modulul timer sunt implementate câteva mecanisme care pun la dispoziția utilizatorului funcții specifice. Mecanismul de comparare permite controlul unor semnale de ieşire; mecanismul de captură permite monitorizarea unor semnale de intrare; numărătoarele interne permit generarea de referințe de timp interne, necesare în bucle de întârziere, multiplexarea diferitelor sarcini software, ş.a. Timer-ul poate fi folosit practic pentru orice funcție de timp, inclusiv generarea unor forme de undă sau conversii D/A simple. Funcțiile oferite de un timer sunt:

1. <u>Captură la intrare</u> (IC - *input capture*) - această funcție se bazează pe posibilitatea de a stoca valoarea numărătorului principal la momentul apariției unui front activ al unui semnal extern. Facilitatea permite măsurarea lățimii unui impuls sau a perioadei unui semnal. Facilitatea poate fi folosită şi ca referință de timp pentru declanşarea altor operații.

2. Comparare la ieşire (OC - *output compare*) - se compară la fiecare impuls de ceas valoarea numărătorului principal cu cea a unui registru. Dacă se constată egalitate, în funcţie de programarea anterioară pot avea loc următoarele evenimente: declanşarea unei acţiuni la un pin de ieşire (opţional), setarea unui flag într-un registru sau generarea unei întreruperi pentru unitatea centrală (opţional). Funcţia este folosită pentru a genera întârzieri sau pentru a genera o formă de undă cu valori dorite pentru frecvenţă şi pentru factorul de umplere .
3. Întreruperi în timp real (RTI – *real-time interrupt*) – într-un sistem există sarcini care trebuiesc executate periodic sau care nu permit depăşirea unui interval limită între doua execuţii. Aceste sarcini sunt lansate ca rutine de tratare a întreruperii generate de timer.
4. COP (*computer operating properly*) watchdog – această funcţie este folosită pentru a reseta sistemul în cazul în care din erori de programare (*bugs*) sau erori în desfăşurarea programului datorate perturbaţiilor mediului, registrul COP nu este accesat într-un interval de timp prescris.
5. Acumulare de pulsuri (*pulse accumulator*) – este funcţia folosită pentru a număra evenimentele ce apar într-un interval de timp determinat sau pentru a măsura durata unui impuls.

Numărătorul liber

În figura 1.10 sunt prezentate elementele de bază din componenţa unui timer, partea de numărător liber.

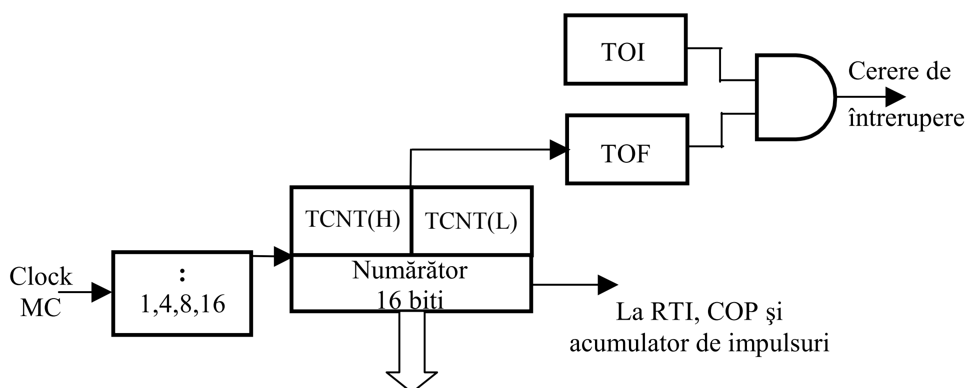


Figura 1.10 Schema bloc a unui timer (MC 68HC11) – numărătorul liber

Numărătorul este pilotat de un clock provenit din clock-ul unităţii centrale divizat cu un factor de regulă programabil. Numărătorul este înscris cu 0 doar la reset, iar în rest este un registru de tip read-only. Când numărătorul tranziţionează din starea FFFFh în starea 0000h, se seteaza flag-ul TOF (*timer overflow*) aflat

într-un registru neoperațional asociat modulului timer. Pentru a autoriza generarea unei întreruperi este necesar să fie setat un flag de autorizare (TOI).

Factorul de divizare a clock-ului este stabilit într-un registru asociat.

Registrul TCNT de 16 biți permite citirea stării numărătorului. Se recomandă citirea numărătorului folosind o instrucțiune care operează pe 16 biți; în acest fel ambii octeți ai numărătorului aparțin aceleiaşi stări de numărare. Dacă se fac două citiri, accesând pe rând partea mai semnificativă şi partea mai puțin semnificativă, cele două informații nu vor aparține aceleiaşi faze de numărare.

Captura la intrare

Unele aplicații necesită cunoaşterea localizării în timp a momentului în care are loc un eveniment. Într-un calculator, timpul fizic este reprezentat prin conținutul unui numărător, iar apariția unui eveniment este reprezentată printr-o tranziție de semnal. Momentul apariției evenimentului poate fi înregistrat prin memorarea stării numărătorului.

Funcția de captură la intrare este implementată de un circuit care are schema bloc reprezentată în figura 1.11.

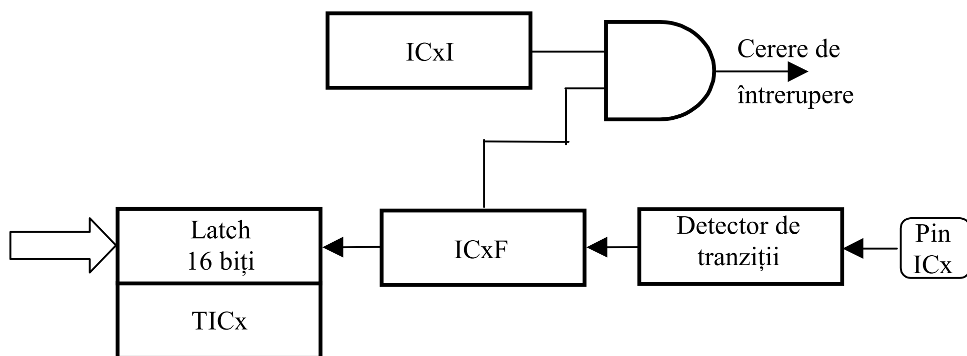


Figura 1.11 Funcția de captură la intrare – schema bloc

Timpul fizic este reprezentat de valoarea conținută de numărătorul liber. La apariția unui eveniment materializat printr-o tranziție la pinul ICx, conținutul numărătorului liber este preluat în latch-ul de 16 biți din blocul de captură. Utilizatorul poate să valideze funcția de captură, poate să aleagă frontul activ al tranziției şi poate să autorizeze generarea unei întreruperi către unitatea centrală la detectarea frontului activ. Dacă un canal de captură nu este validat, pinul de intrare ce îi este asociat poate fi folosit ca intrare de uz general. Întreruperile intrărilor de captură sunt vectorizate, fiecărei întreruperi îi este alocată o poziție bine definită în tabela vecorilor de întrerupere.

Funcția de captură are multe aplicații, câteva din ele descrise pe scurt în continuare:

1. Localizarea momentului apariției unui eveniment - funcția de captură este foarte utilă în cazul în care este necesară cunoaşterea momentelor la care au avut loc evenimentele. Numărul de evenimente monitorizate sub acest aspect este limitat de numărul de canale de captură ale MC.

2. Măsurarea perioadei – pentru măsurarea perioadei unui semnal necunoscut se procedează la determinarea timpului scurs între două tranziții consecutive de acelaşi fel; două fronturi pozitive consecutive sau două fronturi negative consecutive.

3. Măsurarea lățimii unui impuls – Pentru a măsura lățimea unui impuls se măsoara timpul scurs între un front pozitiv şi frontul negativ ce îi urmează. Deoarece numărătorul liber este un numărător pe 16 biți, intervalul maxim măsurabil cu un ciclu de numărare este limitat la de 2^{16} ori periada unui clock (clock-ul numărătorului). Dacă se măsoară intervale mai mari, se va lua în considerare şi numărul de tranziții de la valoarea maximă la 0 a numărătorului liber.

4. Generarea unei întreruperi – Fiecare canal de captură poate genera opțional o întrerupere dacă se detectează la intrarea de captură frontul stabilit a fi activ.

5. Numărarea unor evenimente – considerând că un eveniment este reprezentat de o tranziție de semnal, cu funcția de captură la intrare folosită în combinație cu funcția de comparare la ieşire, pot fi contorizate evenimentele ce apar într-un interval de timp stabilit. La apariția unui eveniment se generează o întrerupere; în rutina de servire a întreruperii se incrementează un contor de evenimente realizând astfel înregistrarea numărului de apariții. Sfârşitul intervalului de contorizare este stabilit cu ajutorul funcției de comparare la ieşire.

6. Referință de timp – aplicația foloseşte în combinație funcția de captură la intrare şi funcția de comparare la ieşire. Dacă utilizatorul doreşte să activeze un semnal de ieşire la un interval dorit după detectarea unui eveniment, se va folosi funcția de captură la intrare pentru a stabili momentul apariției evenimentului de referință. Se va aduna la valoarea numărătorului reținută la captură un număr dorit de cicluri de numărare şi se va transfera această valoare registrului de comparare. Când numărătorul liber atinge valoarea comparatorului se va genera o întrerupere în rutina căreia se va activa ieşirea dorită.

Comparare la ieşire

Scema bloc a componentei care realizeză funcția de comparare este reprezentată în figura 1.12.

Un canal de comparare constă dintr-un comparator pe 16 biți, un registru de comparare pe 16 biți, un circuit de generare de întreruperi, o ieşire de comparare

şi o logică de comandă. Un sistem timer dispune de mai multe canale de comparare (68HC11 are cinci canale de comparare).

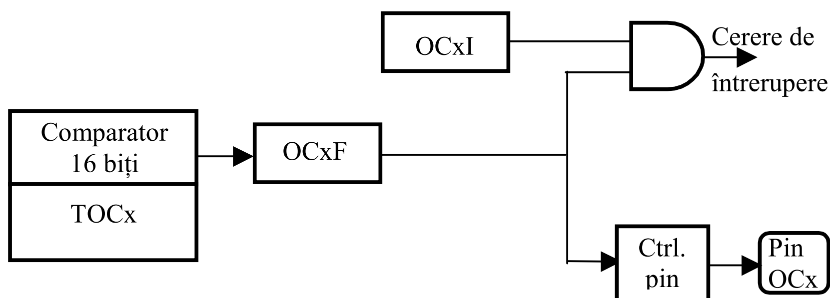


Figura 1.12 Funcţia de comparare la ieşire – schema bloc

Principala aplicaţie a unei funcţii de comparare este întreprinderea unei acţiuni la un moment de timp viitor bine determinat, când numărătorul liber atinge o anumită valoare. Acţiunea întreprinsă ar putea fi comutarea unui semnal, acţionarea unui întrerupător electronic, aprinderea unui LED, etc. Pentru a folosi funcţia de comparare este necesară parcurgerea următoarei secvenţe:

1. se copiază valoarea din numărătorul liber (conţinutul registrului TCNT);
2. se adaugă la copie o valoare determinată de întârzierea dorită;
3. se transferă rezultatul în registrul de comparare.

Acţiunea dorită la pinul ieşirii de comparare se specifică anterior de către utilizator într-un registru asociat modulului. Comparatorul compară valoarea numărătorului liber TCNT cu valoarea din registrul de comparare la fiecare clock. Dacă se constată egalitate, se activează acţiunea specificată la pinul de comparare şi se setează flag-ul asociat. De asemenea, se va genera o întrerupere dacă aceasta a fost validată anterior prin bitul OcxI din registrul pentru validarea întreruperilor. Întreruperile sunt vectorizate; fiecare canal are o a adresă cunoscută în tabela vectorilor de întrerupere.

Întreruperile în timp real – RTI

Funcţia RTI este folosită pentru a genera periodic întreruperi; întreruperile pot fi folosite pentru a comuta între aplicaţii în sisteme multitasking sau pentru a declanşa periodic execuţia unei secvenţe. Perioada apariţiilor întreruperilor în timp real este programabilă. Ca rezoluţie, funcţia de întrerupere nu este la fel de performantă ca şi funcţia de comparare.

COP - Watchdog

COP (*Computer Operating Properly*) este un ceas de gardă, numit *watchdog*, folosit pentru a detecta erorile de program. Folosirea ceasului de gardă este opțională. În cazul în care se foloseşte, programul utilizator trebuie să reseteze periodic un registru COP; dacă nu are loc resetarea, unitatea centrala decide că a apărut o problemă în rularea programului şi resetează sistemul pentru a preveni o desfăşurare necontrolată a programului.

Funcția ceasului de gardă poate fi realizată folosind un canal timer de uz general sau, situația întâlnită în mod obişnuit la MC, ceasul de gardă este un timer dedicat, folosit doar pentru funcția gardă. Structura unui modul watchdog este reprezentată în figura 1.13

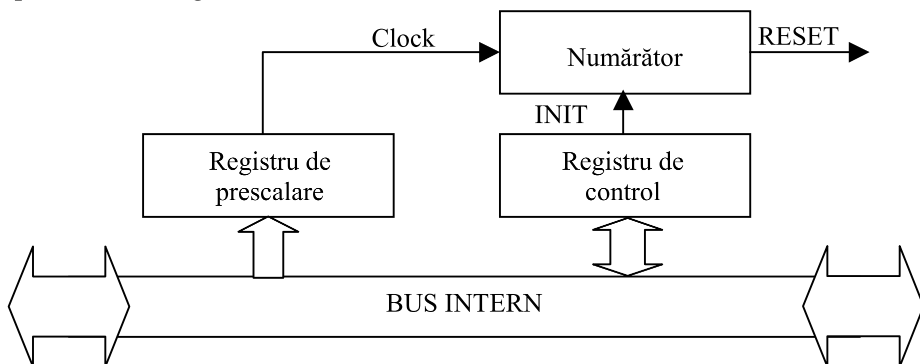


Figura 1.13 Schema bloc a unui modul watchdog

Ceasul de gardă este un timer simplu compus dintr-un numărător al cărui ceas de numărare este programabil printr-un registru de prescalare. Dacă circuitul este activat, numărătorul numără continuu. Dacă numărătorul ajunge la valoarea cea mai mare (FFFFh, dacă este un numărător de 16 biți). se generează un RESET către unitatea centrală. Este sarcina programatorului să scrie periodic în registrul de control un cuvânt care să inițializeze numărătorul. În cazul în care programul a pierdut controlul se va reseta sistemul.

Acumulatorul de impulsuri

Modulul timer poate să dispună de un registru acumulator de impulsuri. În acest registru pot fi numărate evenimente exprimate prin pulsuri sau pot fi numărate chiar pulsuri interne, provenite de la clock-ul unității centrale.

1. <u>Numărare de evenimente</u> – numărătorul de 8 biți este incrementat la fiecare front activ detectat la un pin. Un eveniment poate fi orice, segmente de program, cicluri ale unui semnal de intrare, unități de timp, etc. Pentru a fi numărate în acumulatorul de impulsuri, acestea trebuiesc transformate în tranziții.

2. Numărare gardată – numărătorul de 8 biți funcționează în regim liber de numărare având la intrarea de clock un semnal provenit din clock-ul unității centrale divizat cu un factor constant. O aplicație uzuală în acest mod de funcționare este măsurarea duratei unui puls (impuls singular). Numărătorul este înscris cu 0 la începutul pulsului, iar la sfârşitul pulsului se citeşte valoarea sa.

Module PWM

Un semnal PWM (*Puls Width Modulation*) este folosit în multe aplicații; comanda motoarelor de curent continuu sau comanda surselor de alimentare pot fi amintite ca principale aplicații. În figura 1.14 este reprezentat un semnal PWM.

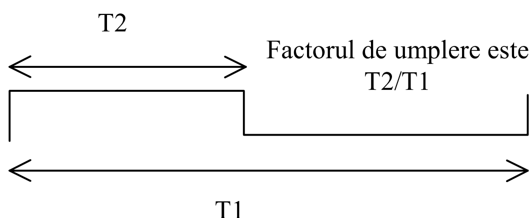


Fig. 1.14 Semnal PWM

Semnalul PWM este un semnal periodic la care se poate modifica în mod controlat factorul de umplere. Modulele timer au resursele necesare generării unui semnal cu factor de umplere variabil. Deoarece, după cum am mai amintit, semnalul PWM este utilizat în aplicații există module timer dedicate acestei funcții; modulele PWM. Un modul PWM poate genera mai multe semnale modulate. Structura unui canal PWM este reprezentată în figura 1.15.

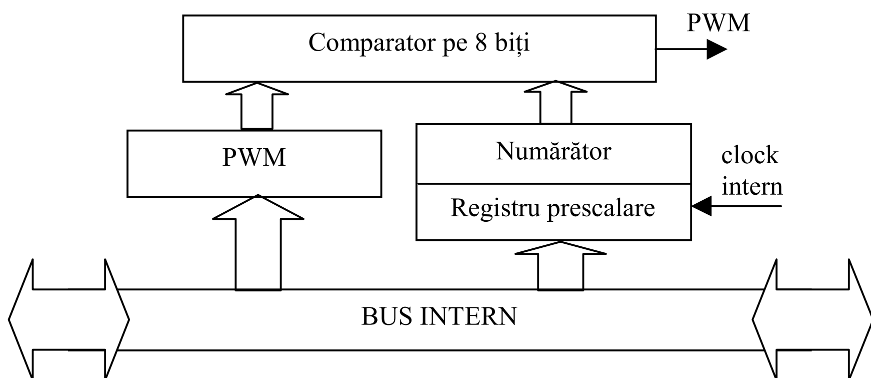


Fig. 1.15 Schema bloc a canalului PWM

Registrul de prescalare generează clock-ul pentru un numărător. Clock-ul de numărare este programabil. Conținutul numărătorului este comparat cu cel al

registrului PWM. Cât timp rezultatul comparației este mai mic sau egal, se va genera un unu logic la ieşirea PWM. Dacă rezultatul comparației este mai mare, se va genera la ieşirea PWM un zero logic. Dacă registrele comparate sunt de opt biți factorul de umplere poate fi între 1/256 şi 1. Un canal PWM odată programat va genera la ieşire semnalul periodic continuu, fără intervenția unității centrale.

Ca şi la celelalte componente funcționale, în ceea ce priveşte global modulele timer (includem şi modulele watchdog şi PWM) vor fi prezentate în încheiere câteva criterii de apreciere:

- numărul de canale timer şi dimensiunea registrului de numărare;
- flexibilitaea timere-lor, mecanisme implementate;
- existența unui ceas de gardă (watchdog);
- numărul de canale PWM şi dimensiunea registrului PWM.

Module A/D şi D/A

Intrările/ieşirile analogice şi convertoarele A/D nu sunt considerate ca parte definită în contextul unui calculator; aceste elemente sunt importante în schimb atunci când considerăm un microcontroller. Prezența modulelor A/D şi D/A în structura unui MC contribuie la "puterea"acestuia în aplicații deoarece interfațarea cu mediul presupune necesitatea de a prelucra sau de a elabora mărimi analogice.

Convertoarele A/D sunt des întâlnite printre perifericele *on-chip*. Convertoarele D/A nu sunt întâlnite în mod uzual printre unitățile componente deoarece sunt relativ uşor implementate în exterior. Un convertor D/A simplu poate fi realizat folosind un timer în mod PWM şi integrând pulsul în exterior cu un simplu circuit RC.

Convertoarele A/D integrate pe chip sunt convertoare cu aproximații succesive sau mai rar, convertoare cu integrare. Însuşirile convertoarelor nu sunt deosebite; sunt convertoare lente în comparație cu cele implementate în circuite independente. Timpii de conversie obişnuiți sunt plasați în intervalul 10μs-25μs. Rezoluția este de 8, 10 sau 12 biți iar precizia de +/-1/2LSB. Modulul de conversie este prevăzut şi cu un multiplexor analogic, astfel sunt disponibile mai multe canale de intrare. Unele MC sunt echipate şi cu circuit de eşantionare/memorare. În cazul în care circuitul de eşantionare/memorare lipseşte, semnalul analogic trebuie menținut constant pe durata unei conversii. Tensiunea de referință necesară convertorului poate fi generată în circuit sau dacă nu, este necesar să fie furnizată din exterior.

În continuare, în figura 1.16 este prezentată o schemă bloc simplă a unui modul de conversie A/D. Circuitul analogic de intrare constă într-un multiplexor analogic, un circuit de eşantionare/memorare şi un convertor A/D cu aproximații succesive. Tensiunea de referință pentru convertor şi masa analogică sunt furnizate din exterior la pini speciali. Clock-ul necesar convertorului este generat intern din clock-ul unității centrale.

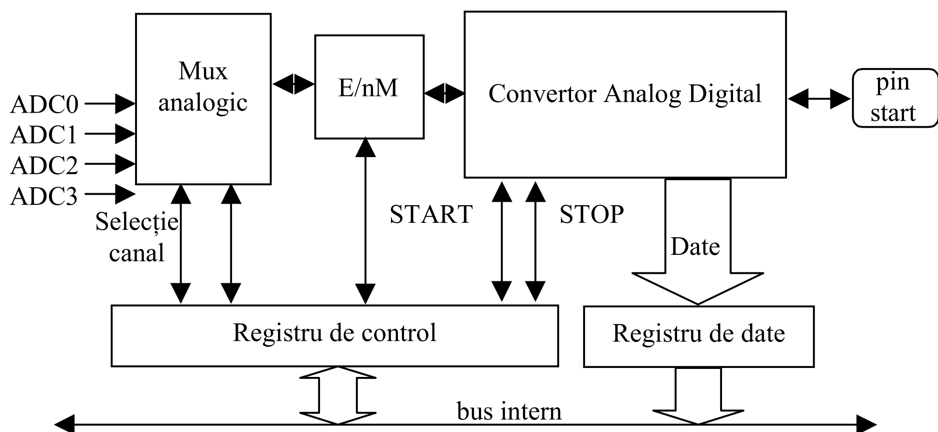


Figura 1. 16 Schema bloc a modului de conversie A/D

Modulul A/D foloseşte un registru de control prin care se selectează canalul de conversie şi modul de lucru pentru circuitul de eşantionare/memorare. Declanşarea şi terminarea conversiei sunt semnalizate cu câte un bit tot în registrul de control. Rezultatul conversiei este stocat în registrul de date. Registrul de date va conține întotdeauna rezultatul ultimei conversii, de aceea acest registru trebuie citit înainte de terminarea următoarei conversii, în caz contrar se pierde informația

Conversia poate fi declanşată intern, considerând bitul asociat din registrul de control, sau din exterior printr-un semnal de comandă aplicat la pin. Declanşarea unei conversii la un moment de timp dorit, fără intervenția unității centrale, se poate realiza folosind un timer. Modulul poate fi programat să execute o conversie a unui canal şi să se oprească. Un alt mod de operare al modulului poate fi conversia continuă a unui canal până la recepționarea unei comenzi de încheiere. Un alt mod de operare este conversia ciclică în care modulul execută pe rând conversia fiecărui canal, după care se opreşte. În acest ultim mod sunt folosite de obicei mai multe registre de date pentru memorarea rezultatelor conversiilor. Rezultatele sunt citite după încheierea unui ciclu. La terminarea unui ciclu de conversiei, opțional, se poate genera o întrerupere către unitatea centrală.

În aplicații de precizie se recomandă ca pe durata conversiei să se comute unitatea centrală în stare inactivă pentru ca aceasta să nu perturbe convertorul.

1.4 SISTEMUL DE ÎNTRERUPERI

Sistemul de întreruperi reprezintă mecanismul ce asigură sincronizarea funcționării unității centrale cu evenimentele. Acest mecanism asigură implementarea unui răspuns *prompt* şi *specific* al sistemului la orice cerere de

întrerupere. Mulți dintre producători au realizat pentru diversele familii de microcontroller-e variante constructive care diferă uneori substanțial unele față de altele, scopul fiind fie acela al simplificării sistemului, fie acela al asigurării unor facilități cât mai adecvate aplicațiilor pentru care au fost proiectate respectivele microcontroller-e.

Încă din anii 1980-1985 s-a conturat un principiu care se confirmă şi astăzi: "fiecare funcțiune complexă cu procesorul ei", respectiv "fiecare funcțiune a unui sistem de comandă cu rutina ei de servire a întreruperilor".

Evenimentele, sunt acele stări din funcționarea sistemului care trebuie avute în vedere la controlul acestuia. Ele trebuie să declanşeze, cu o întârziere minimă, un răspuns adecvat din partea sistemului, răspunsul trebuind să fie descris de către proiectant prin instrucțiuni incluse în cadrul unor rutine de servire a întreruperilor.

În cazul microcontroller-elor, ca de altfel şi în acela al microprocesoarelor, aceste evenimente trebuie privite ca informații de stare a sistemului, informații ce rezultă fie urmare a prelucrării de către UC a datelor, (evenimente ce generează aşa numitele întreruperi interne), fie urmare interacțiunii sistemului cu mediul înconjurător (numite şi întreruperi externe). Acest din urmă aspect este preponderent în cazul microcontroller-elor.

Unitatea centrală a MC analizează în cadrul ultimei perioade de ceas corespunzătoare fiecărui ciclu instrucțiune eventualele cereri de întrerupere postate pe parcursul execuției instrucțiunii curente.

Ca orice sisteme deterministe, sistemele digitale dotate cu microcontroller-e sunt ancorate, în funcționarea lor, într-un spațiu multidimensional ce are drept dimensiuni parametrii proprii şi cei reglați de către sistem ("mărimile de stare") şi timpul. Sistemul de întreruperi este informat asupra evoluției acestora prin canale de transfer a informațiilor generale (porturi, canale temporizatoare/numărătoare) sau specifice (interfețe specializate UART, SPI, SCI, USB, I2C, etc), el dispune de o unitate de analiză a informațiilor ("controller-ul de întreruperi" - CI), implementată intern, conectată via magistralele interne ale MC la unitatea de control a programelor), precum şi de mecanismul prin care opreşte execuția de către UC a programului curent şi realizează declanşarea rutinei adecvate de servire a întreruperilor.

Realizarea stărilor generatoare de cereri de întrerupere este urmată de analiza acestora de către CI. Această analiză presupune următoarele:

- Identificarea sursei generatoare a cererii de întrerupere;
- Determinarea ordinii de servire a cererilor de întrerupere, în cazul apariției simultane a mai multor astfel de cereri.

Ambele etape pot fi implementate atât hard cât şi soft, neexcluzându-se posibilitatea unor implementări mixte.

Identificarea sursei generatoare de întrerupere se face prin corelația ce este stabilită între canalul de transfer al cererii, registrul/registrele de memorare al acesteia şi adresa de memorie la care este plasată adresa rutinei de servire a

întreruperilor sau prima instrucțiune a rutinei de servire a întreruperilor. Această situație este întâlnită la majoritatea microcontroller-elor, dar există şi excepții, cum ar fi microcontroller-ele familiilor PIC "low range" şi "middle range" de la firma Microchip, la care orice cerere de întrerupere generează execuția unei rutine al cărei început este plasat la o adresă unică, fixă de memorie.

Determinarea ordinii de servire, sau analiza priorității cererilor de întrerupere, presupune scanarea în faza de analiză a tuturor canalelor de transfer pentru cererile de întrerupere sau a regiştrelor corelate cu acestea într-o anumită ordine prestabilită hard. Multe microcontroller-e dispun de mecanismul (implementat hard) ce permite "prioritizarea" unora dintre aceste canale, ceea ce implică implementarea unor unități de memorare a priorităților (registre de setare a priorităților pe două sau mai multe nivele) şi instituirea unor reguli de arbitrare. Fiecare dintre aceste nivele va fi scanat distinct, în ordinea descrescătoare a priorităților, majoritatea CI prezentând doar două nivele de prioritate: prioritar şi mai puțin prioritar.

În procesul de scanare, primul canal "activ" - care generează o cerere de întrerupere - va fi cel ce va fi "servit", respectiv va declanşa procesul de execuție a rutinei de servire a întreruperilor corespunzătoare.

Sunt întâlnite atât microcontroller-e care permit întreruperea unei rutine de servire a întreruperilor de către o altă cerere de întrerupere mai prioritară (îndeobşte cunoscut ca sistem de întreruperi "reîntreruptibil"), cât şi microcontroller-e care nu permit o astfel de acțiune.

Mulți producători pun la dispoziția proiectantului câte un flag de validare/invalidarea globală a întreruperilor, ceea ce permite ca utilizatorul să modifice cum doreşte mecanismul de acțiune la apariția unui întreruperi: permite sau inhibă o întrerupere prioritară apărută ulterior uneia mai puțin prioritare, dar pe durata cât aceasta este servită.

Multe microcontroller-e dispun de un mecanism bazat pe flag-uri (fanioane) de validare/invalidare a întreruperilor, acest mecanism putând fi implementat atât la nivelul UC al MC cât şi la / sau la nivelul CI, respectiv la nivelul sistemului de analiză a cererilor de întrerupere. Registrele de memorare a cererilor de întrerupere sunt dublate de registre ce memorează informația de validare a canalelor întrerupătoare ("registre de validare a întreruperilor" sau "registre de mascare a întreruperilor"), un canal corespunzător unei cereri de întrerupere invalidată va fi ignorat în procesul de scanare a cererilor de întrerupere.

Un principiu des aplicat la implementarea acestui sistem constă în aceea că fiecare flag este setat printr-un mecanism hard şi, în general, resetat soft, prin execuția unor instrucțiuni de resetare a sa, ceea ce presupune pe de-o parte, luarea în considerare prin program a stării semnalate, iar pe de altă parte, asigură reanclanşarea mecanismului de întrerupere, pentru situația în care starea respectivă se repetă.

Domeniul de adrese de memorie (SRAM sau PROM) în care sunt plasate instrucțiunile de start a rutinelor de servire a întreruperilor poartă denumirea de "tabelă a vectorilor de întrerupere".

Unele microcontroller-e prezintă o tabelă a vectorilor de întrerupere cu adrese fixe, specifice canalelor întrerupătoare, spre exemplu I8051, Atmel, PIC, altele permit plasarea tabelei vectorilor de întrerupere la adrese de memorie ce sunt stabilite de către proiectant, este cazul, spre exemplu, micocontroller-elor din familia Z80 (Z8413, Z8414, Z8415, etc.)

În prima situație, plasarea la adresă fixă a tabelei vectorilor de întrerupere, poate fi privită ca o restricție în dezvoltarea unor programe de aplicație, dar aceasta este doar o falsă problemă! Practic, noi putem realiza translatarea prin program a tabelei vectorilor de întrerupere la orice adresă de memorie prin simpla introducere a unor instrucțiuni de salt ca prime instrucțiuni corespunzătoare canalelor întrerupătoare. Exemplificăm aceasta pentru familia 8051.

Tabela vectorilor de întrerupere este cuprinsă între adresele: 0003H şi 0023H. Programul de translatare va trebui să se prezinte astfel:

```
            CSEG
            ORG     00H
START:
            JMP     MAIN            ;Adresa de salt la progamul principal
;
            ORG     03H             ;Adresa din tabela vectorilor de întrerupere
                                    ;corespunzătoare întreruperilor externe de
                                    ;stare - canal 0 -
ISR_EXT0:
            JMP     NEW_ADDR_ISR0 ;Noua adresa a rutinei (poate fi plasată în
                                    ;memoria RAM externă)
;
            ORG     0BH             ;Adresa din tabela vectorilor de întrerupere
                                    ;corespunzătoare întreruperilor datorate
                                    ;depăşirii capacității de numărarea a canalului
                                    ;0 numărător/temporizator
ISR_T0:
            JMP     NEW_ADDR_ISR1 ;Noua adresa a rutinei (poate fi ;plasată în
                                    ;memoria RAM externă)
;
            ORG     13H             ;Adresa din tabela vectorilor de întrerupere
                                    ;corespunzătoare întreruperilor externe de
                                    ;stare -canal 1 -
ISR_EXT1:
            JMP     NEW_ADDR_ISR2
;
            ORG     1BH             ;Adresa din tabela vectorilor de întrerupere
                                    ;corespunzătoare întreruperilor datorate
                                    ;depăşirii capacității de numărarea a canalului
                                    ;1 numărător/temporizator
ISR_T1:
            JMP     NEW_ADDR_ISR3
;
;
```

```
                ORG     23H             ;Adresa din tabela vectorilor de întrerupere
                                        ;corespunzătoare întreruperilor provenite de la
                                        ;UART
ISR_SI:
                JMP     NEW_ADDR_ISR4
;
NEW_ADDR_ISR0:
;Aici se vor plasa instrucțiunile corespunzătoare rutinei de servire a întreruperilor de la
;sursa externă corespunzătoare canalului 0.
NEW_ADDR_ISR1:
;Aici se vor plasa instrucțiunile corespunzătoare rutinei e servire a întreruperilor datorate
;depăşirii de numărare la timer 0.
NEW_ADDR_ISR2:
;Aici se vor plasa instrucțiunile corespunzătoare rutinei de servire a întreruperilor de la
;sursa externă corespunzătoare canalului 1.
NEW_ADDR_ISR3:
;Aici se vor plasa instrucțiunile corespunzătoare rutinei e servire a întreruperilor datorate
;depăşirii de numărare la timer 1.
NEW_ADDR_ISR4
;Aici se vor plasa instrucțiunile corespunzătoare rutinei e servire a întreruperilor provenite
;de la UART.
```

Plasând în memoria RAM această tabelă este posibil să realizăm ceea ce se cheamă "servire dinamică a întreruperilor". Acest procedeu de servire a întreruperilor permite implemenarea uşoară a comenzii unui automat secvențial cu un număr finit de stări. (spre exemplu: cazul unei linii automate de montaj sau de turnare, etc). Implementarea servirii dinamice a întreruperilor constă în modificarea succesivă a adreselor la care plasăm rutina de servire a întreruperii (ISR).

În figura 1.17 prezentăm cazul unui automat cu un număr de trei stări distincte, fiecare rutină de servire a întreruperilor (ISR) va rescrie adresa rutinei de servire corespunzătoare evenimentului următor în tabela vectorilor de întrerupere. Este evident că această tabelă trebuie să fie locatată în memoria RAM a sistemului.

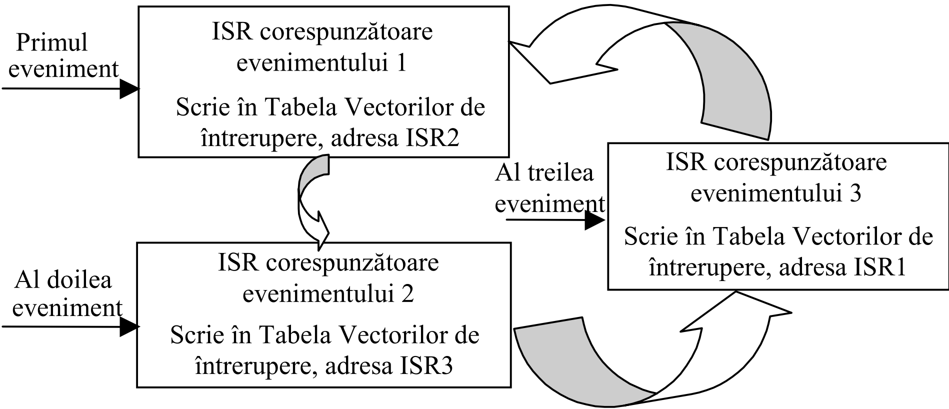


Figura 1.17 Servirea dinamică a întreruperilor într-un automat cu trei stări

Un exemplu tipic în acest sens îl reprezintă tastaturile ce prezintă butoane multi-funcțiune. Să presupunem cazul butonului unic: START/STOP al unui utilaj. Vom exemplifica procedeul de servire dinamică prin programul care comută funcțiunile butonului din START în STOP şi invers. Programul este conceput pentru un microcontroller din familia 8051.

```
                CSEG
                ORG     00H
BOOT:
                JMP     MAIN
ISR_EXT0:
                JMP     BUTON
;S-a considerat că prima linie de întreruperi externe de stare reprezintă canalul de transfer al
;informațiilor corespunzător butonului START/STOP
;
;Instrucțiunea corespunzătoare etichetei BUTON este plasată în memoria RAM a
;sistemului de comandă
BUTON:
                JMP     BUTON_START
;
BUTON_START:
;
                MOV     DPTR,#BUTON+1
;Încarcă adresa din tabela vectorilor de întrerupere
                MOV     A,#LOW(BUTON_STOP)
                MOVX    @DPTR,A
                INC     DPTR
                MOV     A,#HIGH(BUTON_STOP)
                MOVX    @DPTR,A
;Schimbă adresa de salt către rutina următoare în schema de comutare a rutinelor
;Ultima rutină din schemă trebuie să indice drept adresă de salt prima dintre rutine
                RETI
BUTON_STOP:
;
                MOV     DPTR,#BUTON+1
;Încarcă adresa din tabela vectorilor de întrerupere
                MOV     A,#LOW(BUTON_START)
                MOVX    @DPTR,A
                INC     DPTR
                MOV     A,#HIGH(BUTON_START)
                MOVX    @DPTR,A
;Schimbă adresa de salt către rutina următoare în schema de comutare a rutinelor
;Ultima rutină din schemă trebuie să indice drept adresă de salt prima dintre rutine
                RETI
MAIN:
;
;Trebuiesc incluse instrucțiunile corespunzătoare programului principal
END
```

În cazul sistemelor 8051 sau 68xx-68xxx, tabela vectorilor de întrerupere este amplă incluzând variate canale întrerupătoare. Astfel, pentru întreruperile externe de stare se dispune de două linii de intrare INT0# şi INT1#, CI putând fi setat să reacţioneze fie pe front (cel negativ), fie pe nivel logic (cel scăzut -low), pentru canalele numărătoare temporizare dispunem pentru fiecare de adrese de servire corespunzătoare stărilor de depăşire a capacităţii canalului ("overflow"), iar pentru unitatea UART dispunem de un vector unic atât la recepţie cât şi pentru transmisie. În cazul microcontroller-elor PIC (low/middle range) există o singură adresă (plasată la 0004H) pentru întreruperi externe, ceea ce impune pentru proiectant să scrie o rutină de servire în care analizând registrul ce memorează cererile de întrerupere va decide asupra răspunsului la cererea respectivă.

1.5 MANAGEMENTUL PUTERII

Pentru aplicaţiile în care consumul de energie de la sursa de alimentare este critic sunt implementate sisteme prin care pot fi dezactivate temporar parte din modulele componente. Economia de energie este esenţială în cazul alimentării de la baterii. Prin intrarea în starea de economie de energie memoria RAM nu-şi schimbă conţinutul şi ieşirile nu-şi schimbă valorile logice. Sistemul de economie de energie oferă două moduri de operare care diferă prin numărul componentelor dezactivate. Aceste moduri sunt numite *Idle, Power Down* la MC Intel şi *Wait, Stop* la MC Motorola. Intrarea în unul din modurile de operare cu economie de energie se face prin comenzi software. MC Motorola dispun de instrucţiuni care determină intrarea în mod Wait sau Stop. MC Intel intră în mod Idle sau Power Down prin controlul unor biţi din registrul PCON (*Power Control Register*).

În ambele moduri de operare este "îngheţată" funcţionarea unităţii centrale.

În mod Power Down (Stop) este blocat oscilatorul intern ceea ce atrage după sine blocarea tuturor funcţiilor interne. Timer-ul va "îngheţa" şi interfeţele seriale îşi vor opri activitatea. Sistemul de întreruperi, după caz, este dezactivat (MC Intel) sau poate să ramână activ (MC Motorola). Conţinutul memoriei şi al registrelor interne nu este alterat, iar liniile I/O rămân neschimbate. Din această stare se poate ieşi fie doar cu Reset la MC Intel. La MC Motorola se poate ieşi fie cu Reset, fie la sesizarea unei întreruperi externe (sistemul de întreruperi nu este dezactivat). În cazul în care revenirea se face la sesizarea unei întreruperi externe activitatea internă este reluată în aceleaşi condiţii în care a fost abandonată (numărătoarele timere-lor continuă din starea pe care au avut-o la intrare în mod Stop).

În mod Idle (Wait) este îngheţată doar funcţionarea unităţii centrale, celelalte funcţii folosite rămân active (timer-ul, interfeţele seriale şi sistemul de întreruperi). Starea unităţii centrale şi a tuturor registrelor este păstrată cât timp

MC se menține în mod Idle (Wait). Pinii de port păstrează starea pe care au avut-o la intrarea în acest mod. Din modul Idle se poate ieşi prin Reset sau la sesizarea unei întreruperi interne sau externe.

Consumul de energie este minim atunci când sunt activate cele mai puține funcții. Consumul minim de energie se obține în modul de operare Power Down (Stop). În modul Idle (Wait) consumul este redus fața de regimul de operare normală dar este mai mare decât în mod Power Down (Stop). În acelaşi timp, consumul în mod Idle (Wait) este cu atât mai redus cu cât sunt mai puține funcții active la intrare în mod. În mod Power Down (Stop) există posibilitatea unei reduceri suplimentare a consumului prin reducerea tensiunii de alimentare (la unele MC).

OnNow este o inițiativă actuală pentru modernizarea metodelor de economie de energie în sistemele cu microprocesoare. Se intenționează obținerea unui consum de energie (pentru un calculator PC) de o treime din cel actual şi un timp de ieşire din starea inactivă mai mic de 5 secunde. Informații despre această ințiativă se gasesc la adresa www.microsoft.com/hwedev/onnow.htm

Un alt aspect important al alimentării sistemelor construite cu MC este calitatea semnalului furnizat de sursa de alimentare. Dacă nivelul tensiunii de alimentare nu se încadrează într-o plajă permisă este posibil ca programul să funcționeze defectuos, să se abată de la cursul firesc. Pentru a elimina acest risc, pe care nu ni-l putem permite în sistemele de control, în MC sunt implementate module de monitorizare a tensiunii de alimentare. Acestea pot genera un Reset de sistem sau o întrerupere în cazuri considerate limită.

LVR (*Low Voltage Reset*) este un circuit care monitorizează tensiunea de alimentare a unității centrale şi forțează un reset dacă se constată că aceasta este mai mică decât un minim predefinit.

LVI (*Low Voltage Interrupt*) este un circuit care monitorizează tensiunea de alimentare a unității centrale şi generează o întrerupere dacă se constată că aceasta este mai mică decât un minim predefinit. Întreruperea forțează executarea unei rutine de oprire neforțată.

1.6 SCHEMA BLOC A UNUI MC

Cu elementele prezentate până la acest punct schema bloc a unui MC poate fi dezvoltată (în comparație cu cele prezentate în figura 1.1. şi figura 1.2); rezultatul este ilustrat în figura 1.18.

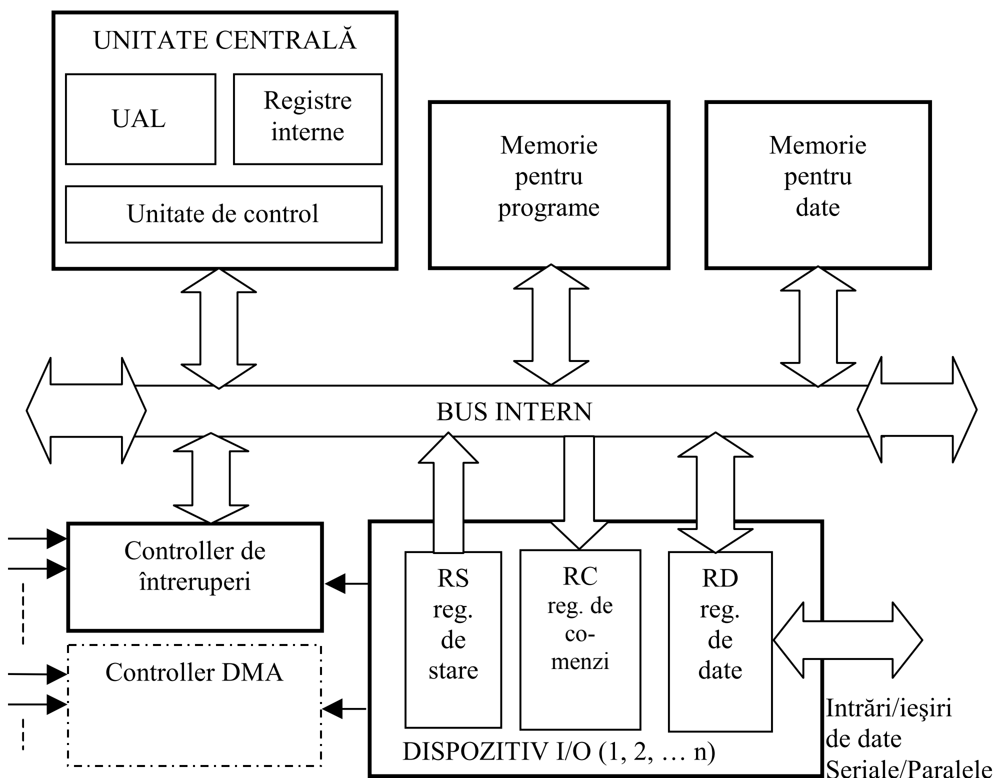


Figura 1.18 Schema bloc generalizată a unui microcontroller

Schema este în continuare o reprezentare generalizată. Se poate observa că un MC este organizat în jurul unei magistrale interne pe care se vehiculează date, adrese şi semnale de comandă şi control între blocurile funcţionale.

Unitatea centrală execută instrucţiunile pe care le primeşte prin magistrala de date din memoria program. Structura Harvard este posibilă şi răspândită la MC pentru că de regulă instrucţiunile sunt stocate în memoria ROM, iar datele în cea RAM. Magistrala de date şi cea de adrese pot fi separate sau multiplexate. Magistralele pot să nu fie disponibile în exterior (Motorola 6805) sau pot fi disponibile în exterior direct (MCS-51) sau multiplexate (MC pe 16 sau 32 de biţi).

Fiecare MC are un controller de întreruperi care admite atât intrări din exterior cât şi de la modulele interne. Unele MC dispun de un controller DMA propriu.

Modulele I/O pot fi seriale sau paralele. Fiecare modul transferă date cu exteriorul prin intermediul registrului de date (RD). Modulul este programat (configurat) de unitatea centrală prin intermediul unui registru de comenzi (RC) şi se poate citi starea modulului prin registrul de stare (RS). Prin RS se pot genera întreruperi către unitatea centrală. Registrele modulelor I/O pot fi văzute de UC ca locaţii de memorie (la familia Motorola) sau ca dispozitive de I/O într-un spaţiu de

adresare separat (MCS 51). Sub numele de dispozitiv I/O, într-o abordare generalizată, sunt cuprinse principalele interfeţe ale MC (timer, canal serial UART) şi linii I/O grupate în porturi paralele de uz general. Acelaşi nume generic poate acoperi şi interfeţele speciale întâlnite în configuraţii particulare de MC (convertor A/D, interfaţă serială sincronă, interfaţă LCD, interfaţă USB, etc).

1.7 FAMILII DE MC

Microcontroller-ele se produc într-o mare diversitate în care există totuşi elemente comune care permit o prezentare sistematică a produsului. Pe baza unui nucleu comun au fost definite familiile de microcontroller-e; nucleul este constituit dintr-o unitate centrală, aceeaşi pentru toţi membrii unei familii, şi o serie de interfeţe şi periferice. Din punct de vedere al programatorului, toţi membrii unei familii folosesc acelaşi set de instrucţiuni, permit aceleaşi moduri de adresare şi folosesc aceleaşi registre. Diferenţa între membrii unei familii constă în primul rând în echiparea chip-ului cu memorie (tip de memorie şi capacitatea memoriei). Alte diferenţe pot fi găsite la frecvenţa de clock pentru unitatea centrală sau în interfeţele on-chip şi perifericele on-chip suplimentare faţă de cel mai simplu reprezentant al familiei. O diferenţă între membrii unei familii poate fi şi modul în care sunt conectate semnalele la pin – respectiv tipul capsulei de prezentare a circuitului integrat. În cadrul acestui capitol, în continuare, sunt prezentate câteva familii de microcontroller-e cu sublinierea însuşirilor caracteristice şi considerând numele producătorului ca fiind unul din elementele reprezentative pentru o familie.

INTEL 404[illegible] fost primul MC apărut pe piaţă, având o structură Harvard modificată, cu 64-256 octeţi de RAM şi este încă folosit în multe aplicaţii datorită preţului scăzut.

INTEL 8051 (MCS-51) este a doua generaţie de MC şi, în prezent, este familia care se vinde cel mai bine; este fabricat şi de mulţi alţi producători. Acest MC are o arhitectură Harvard modificată cu spaţiu de adresare diferit pentru program (<64K din care 4-8K pe chip) şi date (<64K din care 128-256 octeţi pe chip, cu adresare indirectă). Dispozitivele I/O au un spaţiu propriu de adresare. 8051 dispune de un procesor boolean prin care se pot executa operaţii complexe la nivel de bit, iar în funcţie de rezultate se pot face salturi. Pentru 8051 există foarte mult software, atât contra cost cât şi gratuit.

INTEL 80C196 (MCS-96) este a treia generaţie de MC propusă de INTEL. 80C196 este un MC pe 16 biţi care admite un tact până la 50MHz. Acest MC conţine blocuri aritmetice pentru înmulţire şi împărţire, lucrează cu 6 moduri de adresare, convertor A/D, canal de comunicaţii serial, controller de întreruperi cu 8 surse, până la 40 de porturi I/O, generator PWM şi ceas de gardă.

INTEL 80186 şi **8018** sunt MC propuse de INTEL ca versiuni a popularelor microprocesoare 8086 şi 8088 care au echipat primele calculatoare IBM PC XT. Pe chip sunt incluse 2 canale DMA, 2 numărătoare/temporizatoare, controller de întreruperi şi bloc de refresh pentru memoria RAM. Există şi multe versiuni, MC cu consum redus, cu canale seriale etc. Un mare avantaj la folosirea acestor MC este faptul că se pot folosi uneltele de dezvoltare (compilatoare, asambloare etc.) de la PC-uri. Cine este familiarizat cu softul de PC poate trece repede de etapa de învăţare. Un alt avantaj este un spaţiu mare de adresare, specific PC-urilor. INTEl a fabricat şi MC-ul 80386EX, o variantă a procesorului 386 care, la puterea lui 386 mai are în plus canale seriale, canale DMA, numărătoare/ temporizatoare, controller de întreruperi, refresh pentru RAM dinamic, gestionarea alimentării (*Power Management*).

65C02/W65C816S/W65C134S (*Western Design Center*). WDC a creat procesorul pe 8 biţi 65C02 folosit în calculatoarele Apple, Commodore şi Atari urmat de procesorul pe 16 biţi W65C816S şi un MC pe 8 biţi având ca nucleu procesorul de bază 65C02.

Motorola **MC1450** fost primul MC realizat de Motorola. Acest MC în arhitectură RISC avea calea de date de un bit, 16 instrucţiuni şi era furnizat într-o capsulă mică (16 pini) a fost eliminat de pe piaţă de noile MC RISC.

Motorola 680 bazează pe procesorul 6800 şi este asemănător cu 6502 produs de WDC. Are o arhitectură von Neumann în care instrucţiunile, datele, dispozitivele I/O împart acelaşi spaţiu de adresare. Stiva este limitată la 32 de poziţii din cauza indicatorului de stivă pe 5 biţi. Unele MC din această familie includ convertoare A/D, sintetizoare de frecvenţă cu PLL, canale seriale etc.

Motorola 68HC11 eluat şi de TOSHIBA) este un MC popular pe 8 biţi de date şi 16 biţi de adresă, cu o arhitectură ca şi 6805. 68HC11 are inclusă memorie EEPROM sau OTP, linii digitale I/O, numărătoare/temporizatoare, convertoare A/D, generatoare PWM, acumulator de impulsuri, canale seriale de comunicaţii sincrone şi asincrone etc.

Motorola 683xx MC68EC300) sunt super MC de înaltă performanţă bazate pe un nucleu cu arhitecura procesoarelor 68xx, cu o filozofie şi performanţă asemănătoare MC INTEL 80386EX.

PIC (*MicroChip* nt primele MC RISC apărute, cu un număr mic de instrucţiuni (tipic 33, faţă de 8048 care are 90). Simplitatea arhitecturii (Harvard) duce la realizarea unui chip de mici dimensiuni, cu puţini pini, consum redus, viteză mare şi preţ mic. Aceste avantaje au impus MC PIC pe piaţă. Există 3 linii de MC PIC: low range, middle range şi high range (care se disting prin dimensiunea cuvântului de program şi prin facilităţi legate de structură) - PIC16C5xx, PIC16C6xx şi PIC17Cxx, din care linia 16C6xx este cotată cel mai bine pe piaţă.

COP400 (*National Semiconducto* ste un MC pe 4 biţi care are 512o-2K ROM, şi 32x4- 160x4 RAM, în capsule de 20-28 de pini, cu tensiuni de alimentare 2,3V-6,0V. Sunt echipate cu numărătoare/temporizatoare şi magistrală

MICROWIRE. Aceste MC se fabrică în peste 60 de modele echivalente şi sunt primele MC care au un preț sub 0,5USD/bucată.

COP800 (*National Semiconductor*) este un MC de 8 biți care conține MICROWIRE, UART, RAM, ROM, numărătoare/temporizatoare de 16 biți, controller de întreruperi, comparator, ceas de gardă, monitor de tact, generator PWM, transmisie în infraroşu, convertor A/D cu 8 canale cu prescalare (admite şi intrări diferențiale), protecție la scăderea tensiunii de alimentare, mod de aşteptare şi HALT, un trigger SCHMITT şi circuit de trezire cu mai multe intrări (Multi-Input-WakeUp). Eficiența este dată şi de un set puternic de instrucțiuni, majoritatea fiind de un singur octet şi executate într-un singur ciclu. Există şi variante pentru uz militar.

HPC (*National Semiconductor*) sunt MC pe 16 biți de mare performanță, cu o arhitectură von Neumann. Unitatea centrală poate executa înmulțiri şi împărțiri. Conțin în structură funcțiile executate de familia COP800 având în plus canale seriale HDLC (High Level Data Link Control) şi elemente DSP, la o tensiune de alimentare de 3,3V. MC din această familie au multe aplicații în telecomunicații, în sisteme de securitate, imprimante LASER, hard discuri, frâne ABS şi aplicații militare.

Proiectul **PIRANHA** (*National Semiconductor*) este un proiect de realizare a unui MC RISC, primul MC dedicat aplicațiilor integrate. Acest MC are avantajele arhitecturii RISC, deci puține instrucțiuni, simplitate, modularitate.

Z8 (*Zilog*) a fost unul dintre primele MC. Modelul inițial avea UART, numărătoare/temporizatoare, DMA, 40 de linii digitale I/O, controller de întreruperi. Modelul Z8671 avea în ROM un BASIC simplu. Modelul Z86C95 are o structură Harvard, facilități DSP etc. Un avantaj este că se pot folosi pentru extensii unele circuite de interfață din familia lui Z80.

HD64180 (*Hitachi*) este un MC puternic, cu structura şi posibilitățile lui Z80, având în plus 2 canale DMA, canal de comunicații sincrone şi asincrone, numărătoare/temporizatoare şi controller de întreruperi. Unele versiuni includ EPROM, RAM şi PIO (Programmable Input Output). Rulează instrucțiunile lui Z80, dar în mai puține cicluri şi are instrucțiuni în plus. Există variante care funcționează până la 18MHz.

TMS370 (*Texas Instruments*) este similar cu 8051. Conține RAM, ROM (OTP sau EEPROM), 2 numărătoare/temporizatoare de 16 biți, controller de întreruperi, ceas de gardă, generator PWM, convertor A/D cu 8 canale, SCI (port serial asincron), SPI (port serial sincron), comparator, poate executa înmulțiri şi împărțiri. Tactul poate fi până la 20MHZ, cu 5MHz tact de magistrală.

1802 (*RCA*) este un MC mai vechi, cu o structură apropiată de un microprocesor. Este folosit mai ales în aplicații spațiale.

MuP21 (*Forth*) este un MC care are puterea de calcul a unui procesor INTEL 80486 (100MIPS) la un preț mult mai mic. Are integrat un coprocesor video care gestionează o memorie video, deci se poate ataşa direct un monitor TV. Acest MC a fost creat ca să ruleze programul OKAD (program de proiectare VLSI) şi execută acest lucru de 10 ori mai repede ca un 486. Mup21 are 40 de pini.

F21 (*Forth*) a fost conceput pentru aplicații multimedia şi procesare paralelă. Viteza ajunge la 250MIPS datorită structurii care conține procesoare de prelucrare analogică şi procesor de interfață la rețeaua de calculatoare.

Tabelul 1.6 prezintă caracteristicile de bază pentru câțiva reprezentanți din două familii de MC foarte populare; 68HC11 de la Motorola şi 8051 de la Intel.

Tabelul 1.6

Caracteristici principale pentru MC din două familii reprezentative (68HC11 şi 8051)

Circuit	RAM	ROM/ (E)EPROM	Clock	I/O	A/D	Timer
M68HC11A0	256	-	0.476 µs	4x8 1x6	4/8	9
M68HC11A1	256	512 EEPROM	0.476 µs	4x8 1x6	4/8	9
M68HC11A2	-	2048 EEPROM	0.476 µs	4x8 1x6	4/8	9
M68HC11A8	256	8K ROM 512 EEPROM	0.476 µs	4x8 1x6	4/8	9
M68HC11E1	512	512 EEPROM	0.476 µs	4x8 1x6	4/8	9
M68HC11E9	512	12K ROM 512 EEPROM	0.476 µs	4x8 1x6	8	9
M68HC11F1	1024	512 EEPROM	0.476 µs	4x8 1x6	8	9
I 8035	64	-	2.5 µs	3X8	-	2
I 8039	128	-	1.4 µs	3X8	-	2
I 8041	64	1024 ROM	2.5 µs	3X8	-	2
I 8748	64	1024 EPROM	2.5 µs	3X8	-	2
I 8051	128	4096 ROM	1 µs	4X8	-	2
I 8751	128	4096 EPROM	1 µs	4X8	-	2

1.8 CLASIFICAREA MC

Se pot considera multe criterii de clasificare a MC; de exemplu după dimensiunea magistralelor, după interfețele pe care le au incluse în configurație, după aplicațiile în care se folosesc, după furnizor etc. O clasificare sumară a celor mai uzuale MC, după compatibilitatea software, este:

MC cu arhitectură CISC

- Compatibile 8051
- MOTOROLA 68xx
- Compatibile x86
- COP8 (National Semiconductor)
- TMS370 (Texas Instruments)
- ST (Thomson)
- Alte arhitecturi (MC low cost) HITACI - 4biți, Z8 - 8 biți

MC cu arhitectură RISC

- Super H - Hitachi
- PIC - MicroChip
- AVR şi ARM - Atm



2

PROGRAMAREA SISTEMELOR CU MC

2.1 PROIECTAREA PROGRAMELOR DE APLICAȚIE

2.1.1 Generalități 

Dezvoltarea aplicațiilor cu microsisteme presupune pe lângă proiectarea circuitelor electronice aferente acestora şi specificarea în concordanță cu aceasta, prin instrucțiuni, a funcțiunilor pe care sistemul trebuie să le realizeze.

Prin faptul că microsistemele dispun de circuite microprogramabile, structura acestora va fi adaptată aplicației specifice prin rularea programelor de inițializare a sistemului, programe care duc la o configurare particulară, specifică. Deci, una dintre funcțiunile de bază pe care microsistemele dotate cu microcontroller-e trebuie să o implementeze o reprezintă chiar propria configurare -inițializare-, căci având drept suport aceeaşi structură, un sistem realizează noi funcțiuni ca urmare a programării inițiale sau pe parcurs.

O problemă practică ce este pusă la programarea microsistemelor o reprezintă organigrama pe care trebuie să o utilizăm la proiectarea programelor de funcționare. Aceasta variază în raport cu "uneltele" soft avute la dispoziție: asamblor[1], compilator[2], link-editor[3], convertor OBJ-HEX[4], depanator[5], emulator[6], mediu de simulare[7], etc.

---

[1] **Asamblorul** este acel program ce converteşte fişierul sursă -fişier de tip text (ASCII) ce conține instrucțiuni ale microcontroller-ului sub formă de mnemonice, în fişier obiect, respectiv într-un fişier ce conține codurile maşină corespunzătoare instrucțiunilor specificate în cadrul fişierului sursă

[2] **Compilatorul** este acel program ce converteşte fişierul sursă, scris în limbaj evoluat, în fişier obiect.

Metodologia proiectării se bazează pe analiza structurată a aplicației, pe separarea cât mai rațională a funcțiunilor sistemului în raport cu necesitățile de comandă şi cu cele de sincronizare în timp. Se disting ca "stiluri" de programare, stilul de programare ce utilizează extensiv implementarea prin rutine specifice a fiecărei funcțiuni, ceea ce duce la generarea unui program ce prezintă multe apeluri de funcțiuni, utilizează o stivă relativ extinsă, dar care foloseşte eficient memoria sistemului şi stilul de proiectare a programelor care specifică prin macro-uri fiecare funcțiune în cadrul programului principal şi minimizează astfel numărul de apeluri, implicit reduce dimensiunea stivei, dar creşte memoria necesară pentru program.

Evident, ținând cont de varietatea aplicațiilor cât şi de varietatea implementărilor hard, nu este posibilă enunțarea unei "rețete" după care să procedăm! Fiecare aplicație va trebui să fie tratată adecvat, măsura în care implementăm prin rutine, direct în programul principal, sau în rutinele de servire a întreruperilor, funcțiunile sistemului va depinde esențial de coordonata "timp", respectiv, de măsura în care putem îndeplini restricțiile specifice unui sistem de comandă şi control în timp real.

Proiectarea programelor pentru un sistem impune ca proiectantul să considere în strânsă corelație două aspecte: *cel specific procesului* şi cel ce ține de *specificitatea procesorului.*

---

[3] **Link**-editorul realizează cuplarea mai multor module de program memorate în fişiere obiect distincte şi alocă zonele de memorie necesare programelor şi variabilelor utilizate de către acestea. Link-editorul favorizează dezvoltarea structurată a programelor precum şi reutilizarea unor module de program în aplicații distincte.

[4] **Convertorul OBJ-IntelHEX**, realizează conversia din format OBJ (cod maşină), într-un format ce conține: adresa de locatare, numărul de octeți ai liniei respective, tipul datelor (cod sau date), şirul de coduri, suma de control corespunzătoare tuturor informațiilor liniei respective şi terminatorul de linie format din caracterele CR, LF.

[5] **Depanatorul** permite execuția pas cu pas, sau secvență după secvență, a unui program aflat în faza de testare cu scopul de a releva interacțiunea acestuia cu procesul. În general acest program funcționează înpreună cu un sistem de dezvoltare şi dispune de un monitor - programul de bază ce oferă funcțiuni de depanare pe sistemul de dezvoltare-

[6] E**mulatorul** este programul care permite simularea funcțiunilor şi instrucțiunilor microcontroller-ului pe un alt sistem de calcul gazdă. El poate prezenta unele restricții în special în ceea ce priveşte posibilitățile de simulare a întreruperilor. Acest program poate să funcționeze atât pe un calculator gen PC, cât chiar pe microcontroller-ul ce face obiectul aplicației, atunci când acesta o permite. În această din urmă situație, sistemul cu microcontroller are drept consolă chiar un PC, ce este conectat serial la sistemul cu microcontroller. Emulatorul rezident pe circuit permite o simulare avansată a funcționalității sistemului putându-se verifica atât funcționarea programelor cât şi aceea a rutinelor de servire a întreruperilor.

[7] **Mediul de simulare** este utilizat în cazul unor aplicații extrem de complexe, caz în care practic sistemul ce implementează aplicația este dezvoltat cu facilități de depanare. Un astfel de sistem formează un mediu de simulare. El include atât facilități soft (atât pe sistemul gazdă ("host"), cât şi pe cel țintă ("target")), cât şi facilități hard ce țin de existența unor generatoare de semnal programabile, care să ofere excitațiile adecvate.

Deşi varietatea sistemelor este deosebit de largă, totuşi putem considera că, în general, două clase de sisteme sunt preponderente, iar la limită acestea sunt unicele întâlnite în practică, şi anume:

- *sisteme de reglare*, sisteme ce îndeplinesc una sau mai multe funcții scop, care sunt mețînute în limite de variație prestabilite, şi
- *sisteme de control*, respectiv sisteme ce implementează un automat cu un număr finit de stări

Tot aici trebuie să subliniem deosebit de strânsa legătură ce apare între aspectele "hard" ale sistemului şi aspectele "soft" ale acestuia. Această legătură trebuie să fie permanent în atenția proiectantului, el trebuie să-şi "***închipuie***" ***cum şi în ce se traduce execuția fiecărei instrucțiuni şi de asemenea, mai important, să țină cont de sincronizările ce apar în "viața sistemului" între funcționarea unității centrale şi procesul reglat.*** Spre deosebire de programarea în limbaje de înalt nivel sau nivel mediu, în cazul nostru rezultatele programării se traduc prin acțiuni specifice în cadrul sistemului, sistemul de întreruperi jucând un rol esențial. Interacțiunile, în acest caz sunt uneori mai greu de vizualizat, de aceea în timp s-a dezvolat o amplă varietate de unelte de dezvoltare care sprijină proiectantul de soft în realizarea acestuia.

Prin sistem de dezvoltare se înțelege acel sistem ce include procesorul împreună cu memoria aferentă, (SRAM şi EPROM/Flash) şi care dispune de porturi, eventual de posibilitatea de generare de semnale, spațiu pentru "cablarea" unor subsisteme adiționale specifice (interfețe analog digitale şi digital-analogice, driver-e de putere, etc.) şi de programe "monitor[8]" rezidente în memoria fixă a sa, ce implementează dispozitivul logic "consolă" drept legătură cu PC-ul. Aceste programe pot dispune şi de alte rutine, cum ar fi un asamblor/dezasamblor ce permite scrierea sau modificarea "on-line" a programelor ce rulează pe sistem.

Sistemul dispune de "uneltele soft" ce ajută la asamblarea programelor, editarea acestora precum şi a datelor din memoria sistemului de dezvoltare, darea în execuție a programului, oprirea acestuia în puncte prestabilite ("break points"), vizualizarea stării registrelor generale, etc. Toate acestea presupun existența unui dispozitiv logic, "consolă", care asigură controlul sistemului, fie de o consolă propriu-zisă, fie, mai des, de către un PC ce este conectat serial la sistem.

Aspecte specifice de programare apar şi ca urmare a arhitecturii procesorului CISC, RISC sau SISC. Aceste aspecte sunt "*detalii*" extrem de importante în programare, căci ele duc la modificarea stilului de proiectare a programelor.

Să analizăm pe rând aceste aspecte.

Prin analiza necesităților impuse de implemetarea comenzii şi controlului unui sistem, va trebui să înțelegem următoarele.

---

[8] Monitorul este acel grup de programe, rezident în memoria PROM/EPROM a unui sistem de dezvoltare, care dispune de un set minimal de funcțiuni ce ajută la proiectarea şi depanarea pe placă a programelor scrise în cod maşină.

Analiza necesităților de comandă şi control pe care le impune sistemul.

1. Canale digitale de transfer a informațiilor - (vom avea în vedere, numărul acestora, şi modul în care ele pot fi grupate). Este bine să asigurăm corespunzător fiecărei linii de comandă, sau de achiziție digitală câte un bit corespunzător în cadrul unor locații de memorie ce reflectă permanent starea acestora.
2. Canale analogice de transfer al informațiilor - vor necesita alocarea unui număr de linii de ieşire (este cazul convertoarelor digital-analogice), respectiv un număr de linii de intrare (este cazul convertoarelor analog-digitale) ce este funcție de tipul de convertor, respectiv de interfața acestuia cu microsistemul.
3. Necesitățile de control a timpului - (frecvenței, perioadei sau întârzierii) în cadrul sistemului sunt implementate prin utilizarea judicioasă a canalelor temporizatoare.

Fiecare dintre aceste necesități le vom exemplifica în continuare, urmând ca la capitolul de aplicații să concretizăm aceste exemple în programe mai ample.

Spre exemplu: dacă avem de comandat două motoare pas cu pas tetrafazate (MPP) cu *aceeaşi frecvență* de comandă şi în *acelaşi sens*, este evident că vom aloca liniile aceluiaşi port drept linii de comandă ale acestuia. De ce? Căci astfel realizăm două obiective: maximizăm utilizarea liniilor unui port de 8 biți, aşa cum sunt majoritatea, pe microcontroller-ele pe 8 biți, dar totodată simplificăm rezolvarea soft a comenzii, noi dedicând astfel doar o locație de memorie (STATUS_MPP), drept copie a stării liniilor de comandă dedicate motoarelor. Totodată ținând cont că majoritatea controller-elor dispun de instrucțiuni de rotire (dreapta/stânga) a informațiilor, comanda în secvență simplă a MPP nu va presupune decât câteva instrucțiuni:

- Inițializarea contorului de paşi (STEPS1 şi STEPS2)
- Inițializarea locației de stare a MPP-urilor (STATUS)
- Inițializarea timer-ului corespunzător frecvenței de comandă a MPP (SPEED_L, SPEED_H)
- Definirea rutinei de servire a întreruperilor, corespunzătoare canalului temporizator (ISR_T0).

Această rutină va roti informația din byte-ul de stare corespunzător, va copia această informație la port şi în final va decrementa contorul de paşi.

Se vor testa cele două locații ce rețin numărul de paşi STEPS1 şi STEPS2 iar atunci când s-au executat paşii programați, comanda va fi întreruptă.

Prezentăm în continuare acest program implementat pentru microcontroller-e din familia 8051. Instrucțiunile acestui microcontroller sunt prezentate în capitolul curent.

```
                DSEG
0030    STATUS_MPP      DATA    30H
;Locația memorează starea comenzilor pentru MPP (secvență simplă)
0031    STEPS1          DATA    31H     ;Locație nr. paşi MPP1
```

```
0032        STEPS2          DATA   32H    ;Locație nr. pasi MPP2
0033        SPEED_L         DATA   33H    ;Locație low(CT)timer 0
0034        SPEED_H         DATA   34H    ;Locație high(CT)timer 0
                    BSEG
0010        SENS            BIT    10H    ;Bit "sens de rotație MPP"
0011        COMMAND         BIT    11H    ;Bit comandă MPP
;Setarea acestui bit generează inițierea mişcării MPP. Pentru aceasta trebuie să
;inițializăm în prealabil: STEPS1, STEPS2, SPEED_L, SPEED_H şi SENS.
;Bitul SENS este setat la rotirea spre stânga şi resetat la rotirea spre dreapta
                CSEG
                EXTRN    CODE(INIT)    ;Declarație rutina INIT externă
0000            BOOT:
0000 020054         JMP    MAIN
0003                ISR_EXT0:      ;Începutul tabelei vectorilor de întrerupere
0003 02000E         JMP    ISR_T0
000B                        ORG    0BH
000B                ISR_TIMER0:
000B 02000E         JMP    ISR_T0
000E C0D0           ISR_T0:  PUSH   PSW          ;Salvare PSW în stivă
0010 C0E0           PUSH   ACC                   ;Salvare Acumulator
0012 D2D3           SETB   PSW.3 ;Comutare în bancul 1 de registre generale
0014 E531           MOV    A,STEPS1       ;Încarcă în A nr.paşi MPP1
0016 B4000C         CJNE   A,#0,CONT_01  ;Test STEP1=0?
0019 E532           MOV    A,STEPS2       ;Da!
001B B40026         CJNE   A,#0,CONT_02  ;Test STEP2=0?
001E C28C           CLR    TR0                ;Anulare funcționare timer 1
0020 D0E0           END_ISR:  POP    ACC           ;Da! Ieşire ISR
0022 D0D0           POP    PSW                ;Refacere stare registre generale
0024 32             RETI                     ;Ieşire din ISR
0025 E532           CONT_01: MOV A,STEPS2        ;STEP1<>0
0027 B40012         CJNE   A,#0,CONT_03    ;Test STEP2=0?
002A E530           MOV    A,STATUS_MPP    ;Da! STEP1<>0, STEP2=0
002C 540F           ANL    A,#0FH   ;Maschează biții corespunzători MPP 2
002E 1531           DEC    STEPS1    ;Actualizează contor de paşi MPP1
0030 201006         EXEC_STEP:  JB   SENS,LEFT ;Test sens rotație MPP
0033 03             RR     A          ;Roteşte informația de comandă la dreapta
0034 F590           EXEC_BOOTH:   MOV  P1,A ;Scrie comanda către MPP
0036 02004C         JMP    REPRG_T0
0039 23             LEFT:   RL   A ;Roteşte informația de comandă la dreapta
003A 80F8           JMP    EXEC_BOOTH
003C E530       CONT_03: MOV   A,STATUS_MPP ;STEP1<>0, STEP2<>0
003E 1531           DEC    STEPS1
0040 1532           DEC    STEPS2              ;Actualizare informație de paşi
0042 80EC           JMP    EXEC_STEP
```

```
0044 E530          CONT_02: MOV  A,STATUS_MPP ;STEP2<>0, STEP1=0
0046 54F0              ANL     A,#0F0H ;Maschează biții corespunzători MPP 1
0048 1532              DEC     STEPS2
004A 80E4              JMP     EXEC_STEP
004C 85338A        REPRG_T0:      MOV     TL0,SPEED_L
004F 85348C            MOV     TH0,SPEED_H          ;Reprogramează timer0
0052 80CC              JMP     END_ISR
0054                   MAIN:
0054 120000   F        CALL    INIT     ;Execută rutina de inițializare sistem
0057 201102            LOOP:         JB      COMMAND,START_MPP
005A 80FB              JMP     LOOP
005C D28C              START_MPP:     SETB    TR0  ;Validează temporizare T0
005E C211              CLR     COMMAND  ;Anulează comanda0060 80F5
                       JMP     LOOP    ;Ciclează (Aşteaptă!)
          END
```

În cazul în care realizăm controlul unui semnal analogic cu ajutorul sistemului format din microcontroller şi convertor digital-analog de 12 biți având drept interfață o interfață paralelă, vom dedica 12 linii din cadrul a două porturi acestui scop.

Într-un mod analog, putem defini şi operațiile de citire a informațiilor de stare atât cele corespunzătoare unor mărimi analogice cât şi cele corespunzătoare mărimilor digitale. Vom dedica fiecărui semnal analogic locații de memorie având o lățime a cuvântului cel puțin egală cu rezoluția convertorului (1 byte pentru o rezoluție de 8 biți, 2 bytes pentru rezoluții mai mari de 8 biți şi până la 16 biți, respectiv 3 bytes pentru rezoluții cuprinse între 17 şi 24 de biți, etc.)

Deci, numărul de linii dedicate transferului de informații va depinde de interfața convertorului, iar memoria alocată reținerii stării corespunzătoare mărimii va fi mai mare sau egală cu rezoluția convertorului, vorbim de ceea ce îndeobşte se numeşte "alinierea" datelor, respectiv asigurăm uniformitatea dimensiunii locațiilor de memorie dedicate variabilelor sistemului ceea ce uşurează accesul la informație prin folosirea unor indecşi adecvați.

Necesitățile de control a timpului sunt satisfăcute prin dedicarea unor canale temporizatoare/numărătoare. Aici apar cele mai delicate probleme, mai ales în cazul unor sisteme ce controlează procese cu constante de timp mici. În acest caz critic, este necesar să alocăm pentru controlul întârzierii respective un canal temporizator separat, să minimizăm numărul de instrucțiuni ale rutinei de servire a întreruperilor şi să analizăm, iar apoi să setăm cu atenție prioritățile stabilite la arbitrarea întreruperilor. Va trebui ca sistemul nostru să satisfacă cerințelor impuse de teorema eşantionării atât la achiziția de semnal cât şi în comandă, ceea ce presupune ca frecvența minimă de eşantionare să fie mai mare decât dublul frecvenței maxime ce intervine în cadrul procesului reglat.

Nu trebuie să uităm că în cazul unui sistem controlat de către un microcontroller *paralelelismul în acțiune şi interacțiunea* cu procesul se îmbină

continuu, ceea ce indeobşte poartă denumirea de "*concurenţă*". Această proprietate are un rolul esenţial în cazul unui sistem dedicat (încapsulat - "embedded").

Deseori, pentru sisteme relativ lente putem găsi o perioadă minimă de temporizat, rutina de servire a întreruperilor, corespunzătoare acesteia putând fi utilizată pentru controlul mai multor comenzi, ce prezintă constante de timp multipli ai temporizării de bază. Dacă acest semnal de ceas îl folosim pentru analiza stării globale a sistemului, corelat cu el, putem proiecta rutinele unui sistem de operare[9] în timp real, specific. Un program monitor, (supervizor) va putea analiza periodic starea sistemului şi va declaşa task-ul, rutina adecvată stării. În acest caz, construind uneltele minimale şi anume task[10]-uri, rutine[11] şi subrutine[12] vom putea scrie mult mai uşor programe pentru sisteme distincte, asigurând astfel un grad de portabilitate[13] ridicat al aplicaţiilor.

Implementarea programării structurate presupune decelarea acelor acţiuni repetabile şi executate cu parametri diferiţi în mai multe etape din evoluţia sistemului. Ori aceasta presupune scrierea unor rutine, ca grupuri de instrucţiuni apelabile din programul principal.

Un rol esenţial îl joacă şi modul în care realizăm gestiunea resurselor de memorie. Acestea sunt dedicate câtorva scopuri şi anume:

1. Salvează starea sistemului. În acest sens dispunem de zone de memorie accesibile de byte şi pe bit, zonă numită de obicei "vector de stare". Dimensiunea acestei memorii este specifică aplicaţiei.
2. Salvează temporar informaţiile din registrele generale la apelarea rutinelor sau la execuţia rutinelor de servire a întreruperilor, în general implementarea acestei memorii se face utilizând un pointer memorat într-un registru special al unităţii centrale numit "Stack Pointer", mecanismul de acces fiind implementat hard pentru fiecare microcontroller. Respectiva memorie poartă denumirea de "stivă" şi poate fi plasată la orice adresă validă de memorie(internă sau externă microcontroller-ului), dimensiunea ei având un caracter dinamic.
3. Salvează informaţiile ce sunt achiziţionate, transferate sau prelucrate de către sistem formând ceea ce îndeobşte se numeşte "buffer" sau memorie tampon.

---

[9] Prin sistem de operare în timp real înţelegem ansamblul de rutine ce oferă principalele rutine de comandă şi control a sistemului, cu condiţia respectării teoremei de eşantionare atât în comadă cât şi în control

[10] Prin task înţelegem acele unităţi de program executabile independent şi care de regulă implementează o singură funcţiune ce este apelată de către planificatorul (în cazul nostru, programul principal) sistemului.

[11] Rutinele implementează porţiuni mici de program, ce tratează evenimente ce apar în funcţionarea sistemului. Sunt caracterizate prin rapiditate în execuţie şi de regulă sunt asociate unui anumit eveniment decelabil de către planificatorul sistemului de operare sau declaşate de către o cerere de întreruperi.

[12] sunt porţiuni de program apelate de către task-uri, şi pot fi: dedicate, cazul celor activate de către un "anumit" task, comune, cele apelate de către mai multe task-uri şi re-entrante, cele disponibile a fi utilizate de mai multe task-uri aparent simultan, fiind reîntreruptibile.

[13] Prin portabilitate înţelegem gradul de reutilizare a programelor scrise pentru un anume sistem la o clasă largă de sisteme analoge celui iniţial.

Aceasta este organizată după principii ce pot diferi: FIFO prima dată memorată este şi prima dată extrasă, (*First In First Out*) sau ultima dată memorată este prima dată extrasă LIFO (*Last In First Out*), etc. Dimensiunea zonei de memorie este adecvată aplicaţiei şi fluxurilor informaţionale vehiculate. Tot în această categorie putem încadra acele zone de "memorie comună", utilizate de către task-uri diferite ce rulează pe sistem, zonă prin intermediul căreia sunt pasate informaţii de la un task la altul în scopul sincronizării acestora. De asemenea în cazul unor sisteme multiprocesor se implementează stive circulare de mesaje ce asigură arbitrarea accesului procesoarelor la zona de memorie comună, implementându-se aşa numita căsuţă poştală ("Mail Box"), etc. Dimensiunea acestor zone poate fi atât variabilă cât şi fixă.

2.1.2 Instrucţiuni ale MC

Este evident că în cazul programării microsistemelor un rol deosebit de important îl joacă setul de instrucţiuni al microcontroller-ului. Dacă în limbajele de nivel mediu sau înalt, limbajele prezintă un caracter independent de structura procesorului, limbajele maşină sunt limbaje specifice. Aceasta presupune un grad de detaliere mult mai ridicat ceea ce implică şi cunoaşterea registrelor interne şi a flag-urilor indicatoare de condiţie specifice unităţii centrale, cunoaşterea structurii interne, respectiv a memoriei interne şi a modului în care aceasta este organizată, cunoaşterea porturilor de intrare/ieşire şi a canalelor numărătoare/temporizatoare.

Cu toate acestea, reprezentarea grafică a evoluţiei unui sistem capătă o răspândire din ce în ce mai largă, ea oferind o reprezentare generală a algoritmilor de funcţionare ai sistemelor.

Spre exemplu, reprezentarea EBNF (*Extended Backus-Naur Form*) utilizează simbolurile terminale[14] şi neterminale[15] care sunt conectate în diagrame (grafuri) sintactice ce formează un graf orientat. În cadrul acestor grafuri săgeata exprimă o formulare sintactică acceptată. Pot fi întâlnite următoarele reprezentări: succesiunea, ramificaţia sau alternativa şi bucla sau repetiţia. Corespunzător acestor formulări, limbajele de programare ale microcontroller-elor, oferă instrucţiuni (reprezentabile ca "atomi" lexicali) ce implementează traiectoriile definte cu ajutorului grafului de fluenţă. Trebuie menţionat că în setul de acţiuni se disting câteva clase de instrucţiuni prezente la toate microcontroller-ele, şi anume:

1. Instrucţiuni ce pot fi succesive (pot fi înşirate unele după altele) şi care realizează în general fie un transfer de informaţie, fie o operaţie artmetico-logică, fie o altă acţiune la nivelul unităţii centrale.

---

[14] Simbolurile terminale sunt acei atomi lexicali care nu mai pot fi descompuşi în continuare.

[15] Simbolurile neterminale sunt simboluri rezultate ca urmare a compunerii unor simboluri terminale.

2. Instrucțiuni de decizie (test), care implementează o ramificație a grafului, în general, la nivelul limbajului maşină, aceastea sunt instrucțiuni ce implementează două alternative (deci sunt de tip binar). În cazul microcontroller-elor nu întâlnim decât bifurcații.
3. Instrucțiuni de tip repetitiv, sau instrucțiuni ce implementează o buclă de program şi care sunt reprezentate prin instrucțiuni de test ce se execută fie la începutul, fie la sfârşitul unui bloc de program

Vom exemplifica pentru două familii de microcontroller-e (Intel 8051 şi PIC -"middle range") cele mai sus afirmate, urmând ca în prezentarea de detaliu a familiilor de procesoare să tratăm exhaustiv aceste instrucțiuni[16].

Familia 8051		Familia PIC "middle range"	
Mnemonic	Acțiunea	Mnemonic	Acțiunea
MOV A,<src>	A <- <src>	MOVF f,d	d=0 W <- <f>
MOV <dest>,A	<dest> <- A		d=1 W <- <f>
MOV <dest>,<src>	<dest> <- <src>	MOVWF	<f> <- W
MOV DPTR,#data16	DPTR <-data16	MOVLW	W <- data8
PUSH <src>	INC SP,@SP <- <src>		
POP <dest>	DEC SP,<dest> <- @SP		
XCH A,<byte>	A <-> <byte>		
XCHD A,@Ri	A <-> <@Ri>		
MOVX A,@Ri	A <- @Ri		
MOVX @Ri,A	@Ri <- A		
MOVX A,@DPTR	A <- @DPTR		
MOVX @DPTR,A	@DPTR <- A		
MOVX A,@DPTR	A<-@DPTR		
SWAP A	A[high] <-> A[low]	SWAP f,d	d=0f[high] <->f[low]
ANL A,<byte>	A <- <byte>	ANDWF f,d	d=0 W<-W * <f>
ANL <byte>,A	<byte> <- A		d=1 f<-W * <f>
ANL [17] < byte > <-data8	<byte><- data8	ANDLW[18] k[19]	W <- W*k
ADD A,<byte>	A<-A+<byte>	ADDWF f,d	d=0 W<-W+f
ADDC A,<byte>	A<-A+<byte>+C[20]		d=1 f<-W+f
		ADDLW[21] k	W <- W+k

[16] Abrevieri folosite la descrierea instrucțiunilor pentru procesoarele familiilor 8051 şi PIC middle range:
A registrul acumulator; W registrul de lucru (acumulator); <src> registru sursă de informație; <dest> registru destinație al informațiilor; DPTR Data Pointer (8051); @ indicator pentru adresare indirectă la memoria externă; d flag indicator al direcției; f regsitru f din "File Register" la PIC; data16 este o constantă numerică pe 16 biți; data8 constantă numerică pe 8 biți; SP registru Stack Pointer (8051);

[17] Abrevieri (continuare): ANL este AND logic, analog funcționeză şi instrucțiunile ORL (OR logic) şi XOR (SAU-EXCLUSIV) logic.

[18] ANDWF/ANDLW execută funcția logică AND între operanzi, analog avem IORWF/IORLW execută funcția logică OR, iar XORWF/XORLW execută funcția logică SAU-EXCLUSIV

[19] k constantă numerică pe 8 biți

[20] Carry Flag sau fanionul indicator al transportului

[21] ADDWF execută funcția aritmetică adunare între operanzi, analog avem SUBLW pentru scădere şi respectiv ADDLF/SUBLW pentru adunare/scădere data pe 8 biți

Familia 8051		Familia PIC "middle range"	
Mnemonic	Acțiunea	Mnemonic	Acțiunea
MUL AB	A[low]<- AxBB[high]<-AxB		
DIV AB	A<- int(AxB) A<- mod(AxB)		
RR A	A_0->A_7 A_7->A_6	RRF	A_0->A_7 A_7->A_6 ....
RRC A	A_0->C C->A_7		
RL A	A_7->A_0 A_0->A_1	RLF	A_7->A_0 A_6->A_7 ....
RLC A	A_7->C A_6->A_7		
JMP addr[22]	PC[23] <-addr	GOTO addr	PC <-addr
CALL addr	@SP <- PCPC <- addr	CALL addr	@SP <- PCPC <- addr
RET	PC <- @SP	RETURN	
RETI[24]	PC <- @SP	RETFIE	
		RETLW k	PC<-@SP, W<-k
SETB b[25]	b <- 1	BSF f,b	f_b <-1
CLR b	b <- 0	BCF f,b	f_b <-0
INC A	A <-A+1	INCF	f <- f+1
INC DPTR	DPTR<- DPTR +1		
DEC A	A <- A-1	DECF	f <- f-1
DEC < byte>	Byte<-byte-1		

Instrucțiunile mai sus prezentate fac parte din prima categorie. Lista completă de instrucțiuni este detaliată pe CD la capitolele corespunzătoare celor două microcontroller-e. Urmează instrucțiunile de decizie:

Familia 8051		Familia PIC "middle range"	
Mnemonic	Acțiunea	Mnemonic	Acțiunea
JB addr_rel	b=1 =>PC<-PC+addr_rel	BTFSC f,b	f_B=0 =>PC<-PC+2 Salt peste instr. următoare
JNB addr_rel[26]	b=0 =>PC<-PC+addr_rel	BTFSS f,b	f_B=1 =>PC<-PC+2 Salt peste instr. următoare
JZ addr_rel	A=0 =>PC<-PC+addr_rel		
JNZ addr_rel	A≠0 =>PC<-PC+addr_rel		
JC addr_rel	C=1 =>PC<-PC+addr_rel		
JNC addr_rel	C=0 =>PC<-PC+addr_rel		
JBC bit,addr_rel	b=1 =>PC<- PC+addr_rel,b <- 0		

Ultimul tip de instrucțiuni, cel de ciclare este ilustrat doar prin câteva exemple:

---

[22] addr reprezintă adresa de salt (adresă absolută)
[23] PC este Program Couter-ul (pointer-ul de instrucțiuni)
[24] Întoarcere din rutina de servire a întreruperilor implică rearmarea sistemului de răspuns la cererile de întrerupere
[25] b este numărul bit-ului care va fi setat/resetat
[26] Abrevieri (continuare): addr_rel repezintă adresa relativă de salt specificată în instrucțiune

Familia 8051		Familia PIC "middle range"	
Mnemonic	Acțiunea	Mnemonic	Acțiunea
CJNE A,<byte>,addr_rel	A≠<byte> =>PC <-PC+addr_rel		
CJNE <byte>,#data8,addr_rel	<byte>≠data8 => PC<-PC+addr_rel		
DJNZ <byte>,addr_rel	byte <- byte-1,PC<-PC+addr_rel	DECFSZ f,d	f<-f-1, d=0 =>W <- f f=0 =>PC<-PC+2
		INCFSZ f,d	f<-f+1, d=0 => W <- f f=0 =>PC<-PC+2

Menționăm în încheierea acestei enumerări a instrucțiunilor că în cazul microcontroller-elor nu există instrucțiuni de intrare/ieşire, căci porturile microcontroller-ului sunt "mapate"[27] în spațiul de adrese de memorie intern. Ce rezultă din aceasta? Accesul la elementele externe microcontroller-ului se face prin instrucțiuni având adrese relative la porturile acestuia, adică sunt instrucțiuni de transfer în cadrul memoriei interne a microcontroller-ului.

Proiectarea programelor de funcționare ale unui sistem trece prin mai multe etape ce sunt ilustrate în figura 2.1. Această figură nu prezintă prima şi în acelaşi timp una dintre cele mai importante etape ale proiectării, care constă în analiza sistemică a procesului sau sistemului a cărui comandă se doreşte a fi implementată.

În cadrul acestei etape trebuie să fie evidențiate:

1. Funcțiunile sistemului, lățimea canalelor de comunicație nesesare transferului infomațiilor, precum şi locațiile de memorie țintă (destinație) a acestora
2. Identificarea acelor canale de transfer a informațiilor care asigură sincronizarea UC cu procesul condus
3. Analiza priorităților ce trebuie să fie atribuite canalelor de transfer la tratarea informațiilor pentru arespecta teorema eşantionării
4. Se evidențiază similitudinile ce apar în funcționarea sistemului sunt grupate în raport cu criterii specifice, funcționale canalele de transfer a informațiilor şi sunt conturate task-urile, rutinele şi subrutinele ce implementează funcțiunile sistemului de comandă şi control.
5. Sunt analizate variatele posibilități de implementare hard a aplicației, respectiv sunt dedicate canalelor informaționale liniile porturilor, canalele temporizatoare, celelalte resurse diferitelor necesități ale sistemului

În final trebuie să subliniem că procesul de proiectare este un proces iterativ, care presupune parcurgerea succesivă de mai multe ori a etapelor de proiectare, pentru a se obține o implementare a aplicației optimală.

[27] Prin "mapare" înțelegem definirea unei funcțiuni surjective ce are drept mulțime de definiție mulțimea porturilor şi celorlalte dispozitie periferice şi ca domeniu al valorilor un subdomeniu din spațiul de adrese corespunzător memoriei interne a microcontroller-ului.

Vom descrie pe rând care sunt elementele necesare şi ce rezultă la parcurgerea fiecărei etape în cadrul procesului de proiectare a programelor de aplicație în cazul utilizării uneletelor specifice procesoarelor 8051.

Asamblarea fişierelor sursă scrise în limbaj maşină şi salvate ca fişiere

EDITAREA FIŞIERELOR SURSĂ (asamblare) (FORMAT ASCII)

EDITAREA FIŞIERELOR SURSĂ (în limbaj C) (FORMAT ASCII)

BIBLIOTECI (DIVERSE)

ASAMBLAREA FIŞIERELOR SURSĂ

COMPILAREA FIŞIERELOR SURSĂ

LINK-EDITAREA

CONVERSIA FIŞIERELOR OBIECT ÎN FIŞIERE FORMAT INTEL-HEX

TRANSFERUL FIŞIERELOR PE SISTEMUL DE DEZVOLTARE ŞI DEPANAREA APLICAȚIEI

SCRIEREA PROGRAMULUI DE FUNCȚIONARE ÎN MEMORIA EPROM/FLASH A SISTEMULUI

Figura 2. 1 Etapele proiectării programelor pentru un sistem dedicat

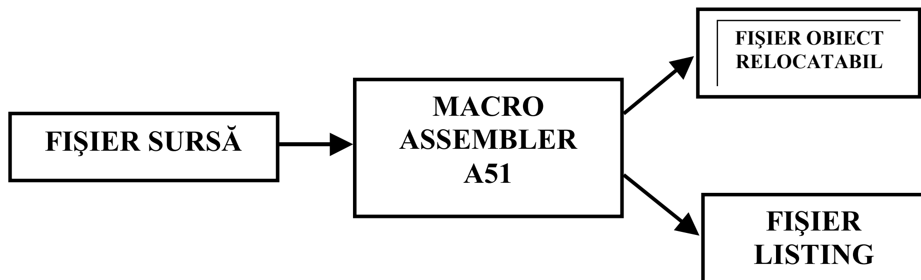


Figura 2.2 Rolul macro/asamblorului în generarea fişierelor obiect

ASCII este descrisă în figura 2.2.

În cazul utilizării unor module de program scrise în C, fişierele sursă vor fi compilate rezultând în urma compilări aceleaşi fişiere ca şi în cazul asamblării.

În figura 2.3 prezentăm care sunt fişierele de intrare şi cele de ieşire în cazul utilizării programului link-editor

După "legarea" utilizând link-editorul a modulelor de program rezultă module obiect absolute care vor fi cu ajutorul debbuger-ului, sau al unui simulator sau al unui emulator, încărcate şi testate pentru a se evidenția intercorelațiile ce se stabilesc între programe şi sistemul controlat.

Pentru transferul fişierelor fie către sistemul de dezvoltare, fie către un programatorul universal este necesar să obținem plecând de la fişierul obiect absolut fişiere în formate transferabile. Această funcție este îndeplinită de convertorul OHS51. (vezi figura 2.4)

Programul LIB51 ce face parte din modulul de programe specifice microcontroller-elor 8051 şi el permite generarea unor biblioteci cu module de

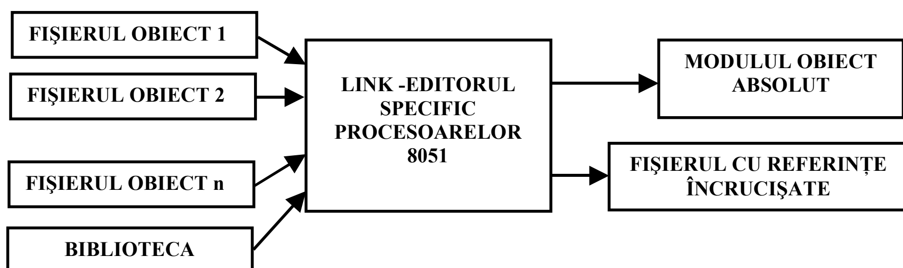


Figura 2.3 Rolul link-editorului în generarea fişierelor obiect absolute

program, adăugarea şi extragerea unor module din bibliotecă.

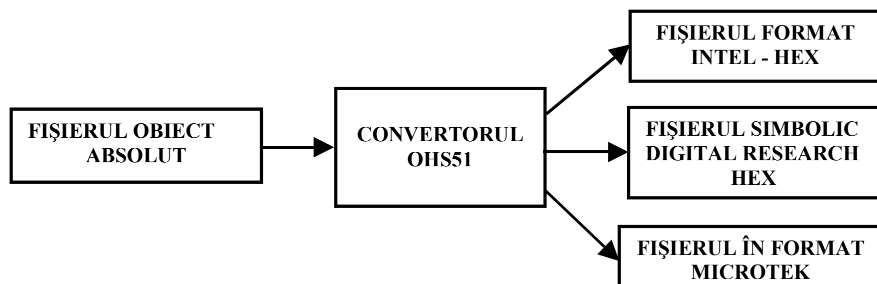


Figura 2. 4 Rolul convertorului object - IntelHEX

2.1.3 Instrumente software de proiectare: MC 8051

Structura generală a unei liniii de program scris în limbaj de asamblare are forma:

[Etichetă**:**] mnemonic8051 [operand][,operand] [;comentariu] <CR><LF>[28]
drept separator de câmp sunt utilizate caracterele spaţiu sau tabulatorul.
Prin etichetă înţelegem şirul de caractere ASCII ce se sfârşeşte prin caracterul ":".
Operanzii pot fi clasificaţi astfel:

- Simboluri speciale de asamblare (nume de registre sau flag-uri)
- Simboluri ale programului (Segment, extern, local, public)
- Adrese indirecte
- Adrese interne ale microcontroller-ului
- Adrese ale biţilor semnificativi ai microcontroller-ului
- Adrese de interes din cadrul programului (în general adrese de salt)

Exemple:
```
START:                          ;Aceasta este o etichetă
        JMP   MAIN              ;Aceasta este o instrucţiune de salt
LOOP:
        DJNZ  R7,LOOP
;R7 este simbolul corespunzător registrului R7, LOOP este simbolul corespunzător
;adresei de salt (acesta este un comentariu)
```

Utilizarea etichetelor nu este opţională atunci când la adresele corespunzătoare acestora se face referire în program. Ele sunt de asemenea utile pentru a da o mai mare lizibilitate programelor. Împreună cu comentariile oferă un real ajutor atât pentru proiectant cât şi pentru cel ce analizează ulterior programele.
Numerele care sunt utilizate în cadrul programelor sursă pot fi scrise în următoarele formate:

Baza	Sufix	Caractere acceptate	Exemple
Hexazecimală	H, h	0,1,…,9,A,B,C,D,E,F	1234H, 0E7H[29], 0A345h
Zecimal	D,d	0,1,2,3,4,5,6,7,8,9	1234, 1234D,023d
Octal	O,o	0,1,2,34,5,6,7	123O, 256O,111Q,125q
Binar	B,b	0,1	11011101B

Şirurile sunt expresii ce includ maximum două caractere încadrate între apostrof-uri.

Exemple: 'A' evaluat ca 0041H
'AB' evaluat ca 4142H
'ab' evaluat ca 6162H, etc.

[28] <CR> reprezintă -returul de car ("Carriage Return"), iar <LF> sfârşitul de linie ("Line Feed")
[29] Caracterele scrise în format hexazecimal care încep printr-o cifră exprimabilă ca un alfanumeric vor fi scrise cu 0 în faţa acesteia. Acest caracter este ignorat la asamblare.

Simboluri: reprezintă valori numerice, adrese sau nume de registre ce permit mai buna lizibilitate a programelor. Atributele ce pot fi date simbolurilor sunt:
TYP: cu valorile **CODE, DATA, BIT** sau **TYPELESS** sau **REGISTER**
SEGMENT: orice simbol conține şi untip de segment.
SCOPE: exprimă validitatea simbolului, ea poate fi **local** (valabilă doar în rutina sau segment de program în care este specificată, **public**, caz în care în toate modulele de program poate fi referit şi este recunoscut, **external,** caz în care definirea simbolului este realizată într-un alt fişier sursă sau **address symbol,**
VALUE: reprezintă valoarea numerică corespunzătoare simbolului, aceasta depinde de adresa tipului de simbol respectiv.
CHANGEALE: Aceste simboluri pot fi redefinite prin intermediul directivei **SET**. Simbolurile care nu au fost definite utilizând directiva **SET** *nu pot fi redefinite* Asamblorul A51.EXE admite drept nume de simboluri şirurile de caractere de până la 31 de caractere, primul trebuind să fie obligatoriu un caracter alfanumeric: 'A','a',…'Z','z'. Celelalte caractere pot fi orice caracter alfanumeric sau numeric.

Asamblorul dispune de anumite simboluri rezervate, respectiv de nume ce definesc resurse ale microcontroller-ului respectiv. Iată câteva dintre acestea: A =acumulatorul, R0,R1,..R7 sunt registre generale ale bancului curent, DPTR este simbolul corespunzător registrului de adresare "Data Pointer", C este flag-ul de transport (Carry Flag), AB sunt registrele A, respectiv B utilizate la înmulțire şi împărțire, iar AR0 la AR7 sunt registre generale ale bancului de registre curente.

Operatorii aritmetici recunoscuți sunt:

Operatorul	Exemplu	Semnificația
+, -	+5 +1 +0Ah	Semnul numărului
+,-	2+10-1	Reprezintă operatorul aplicat
*	1234H*5	Repezintă operatorul de multiplicare
/	45/9	Repezintă operatorul de împărțire
MOD	17 MOD 5	Reprezintă operatorul de împărțire întreagă
^	2^3	Reprezintă operatorul de exponențiere
( )	(2+5)*34	Precizează precedența operațiilor

Operatori binari utilizați sunt:

Operator	Exemplu	Semnificație
NOT	NOT 5	Complementul lui 5
HIGH	HIGH 1234H	Selectează partea mai semnificativă, respectiv 12H
LOW	LOW 1234H	Selectează partea mai puțin semnificativă, respectiv 34H a numărului 1234H

Operator	Exemplu	Semnificație
SHR, SHL	2 SHL 4	Exprimă deplasarea spre dreapta, respectiv stânga a biților numărului 2 cu 4 (ranguri)
AND	0FEH AND 14H	Reprezintă SI -ul logic între cei doi operanzi
OR	0FEH OR 14H	Reprezintă SAU -ul logic între cei doi operanzi
XOR	0FEH XOR 14H	Reprezintă SAU-EXCLUSIV între cei doi operanzi

Operatorii relaționali sunt acei operatori ce compară doi operanzi. Rezultatul comparației este ADEVĂRAT sau FALS, valori ce sunt returnate în urma efectuării operației. Dintre aceşti operatori amintim:

Operator	Exemplu	Semnificație
>= ,GTE	8>=17	Mai mare sau egal. Valoarea returnată: FALS
<= ,LTE	8 LTE 45	Mai mic sau egal. Valoarea returnată: ADEVĂRAT
<> , NE	8 NE 45	Nu este egal. Valoarea returnată: ADEVĂRAT
= , EQ	8 EQ 45	Egal. Valoarea returnată: FALS
< , LT	8 LT 45	Mai mic decât. Valoarea returnată ADEVĂRAT
> , GT	8 GT 45	Mai mare decât. Valoarea returnată FALS

Precedența operatorilor este cea cunoscută şi la alte limbaje de programare, parantezele, apoi operatorii unari, adunarea şi scăderea, înmulțirea şi împărțirea, deplasarea, operatorii logici şi în final operatorii relaționali.

Datele imediate, respectiv valorile numerice incluse în cadrul instrucțiunii sunt reprezentate cu având caracterul '#' în față. Exemple:

```
MOV   A,#0ABH       ;Încarcă în A valoarea ABH, respectiv 171D
MOV   DPTR,#8000H   ;Încarcă valoarea 8000H în DPTR
ANL   A,#128
;Realizează operația ŞI logic între valoarea memorată în acmulator şi valoarea
;128D
MOV   R0,#DAT       ;Încarcă în registru R0, valoarea simbolului DAT
```

Adresarea datelor poate fi făcută în două moduri distincte:

1. Direct caz în care informația de adresă este specificată chiar în codul maşină respectiv
2. Indirect, caz în care simbolul inclus în codul maşină reprezintă adresa la care este memorat operandul utilizat de către instrucțiune. Pot fi utilizați: registrele R0 şi R1 sau registrul DPTR. Spațiul de adrese este funcție de tipul de microcontroller.

Exemplele incluse pe CD detaliază locațiile şi adresele de memorie inclusiv cele corespunzătoare simolurilor corespunzătoare fanioanelor microcontroller-elor din familia 8051.

<u>Directivele de asamblare</u> admise sunt prezentate în enumerarea de mai jos:

SEGMENT permite declararea unui segment ce poate fi relocatat. Sintaxa este:

Nume_segment_relocatabil SEGMENT Tip_segment [Tip_relocatabil]
EQU permite specificarea unei valori pentru o variabilă utilizată în faza de asamblare
SET permite specificarea unei valori pentru o variabilă utilizată în faza de asamblare, valoare ce poate fi ulterior modificată
DATA reprezintă zona de memorie internă având adresele între 0 şi 127
IDATA reprezintă zona adresabilă indirect din memoria internă a microcontroller-ului 0 la 127 sau 0 la 255 inclusiv.
XDATA reprezintă zona adresabilă pe byte situată în memoria externă microcontroller-ului
BIT reprezintă zona adresabilă pe bit situată între 20H şi 27H inclusiv
CODE reprezintă spaţiul alocat codului (programelor executabile)

Rezervarea spaţiului de memorie este realizată prin intermediul dispoziţiilor:

[Nume_bloc_memorie:] DS Dimensiune bloc de memorie
[Nume_variabilă_byte:] DB Valoare de iniţializare byte
[Nume_variabilă_cuvânt:] DW Valoare de iniţializare cuvânt
[Nume_variabilă_bit:] DBIT Valoare bloc biţi

Dispoziţiile ce pot fi incluse se pot adresa şi link-editorului, ele trebuind să fie specificate în programul sursă. Acestea sunt:

PUBLIC directivă ce declară o variabilă sau bloc de memorie "vizibilă" în toate modulele de program sursă.
EXTRN directivă ce precizează faptul că variabila specificată se află în alt modul de program, implicit alt fişier sursă. Se precizează tipul acestei variabile ce poate fi numele unei rutine,deci este de tipul CODE sau o variabilă din memorie, deci DATA.
NAME directivă ce defineşte numele unui modul de program

Dintre dispoziţiile de control a asamblării menţionăm:

ORG adresă Dispoziţie ce specifică adresa de la care se locatează codul scris în continuare
END Indică sfârşitul unui modul de program
RSEG Indică începutul zonei registrelor generale (000H)
CSEG Indică începutul memoriei de program (000H)
DSEG Indică începutul zonei memoriei de date (memoria internă - 000H)
XSEG Indică începutul zonei de memorie externe a microsistemului
ISEG Indică începutul zonei de memorie interne adresabile indirect
BSEG Indică începutul zonei adresabile pe bit
USING Specifică bancul de registre generale utilizat

Dăm în continuare doar un exemplu ce ilustrează cum funcţionează câteva dintre aceste directive.

```
LOC  OBJ  LINE    SOURCE
0020      1       VAL      EQU   20H    ;Iniţializează variabila byte VAL cu 20H
0010      2       VAL1     EQU   16     ;Iniţializează variabila byte VAL1 cu 16
          3                DSEG
```

```
0000    4    DS          10H              ;Rezervă 15 bytes în zona RAM internă
0010    5    REG23:      DS      11H
;Rezervă blocul REG23 de 17 bytes în zona RAM
0021    6    BITI:       DS      01H      ;Rezervă 1 byte cu numele BITI
0030    7    VAL2        SET     VAL+VAL1      ;Ințializează variabila VAL2
;funcție de variabilele VAL, VAL1. Variabila poate fi redefinită în program
REG     8    COUNTER  SET     R0
;Defineşte variabila COUNTER ce poate fi redefinită în program
         9             BSEG     AT      BITI+8
;Setează adresa de la care definim zona utilizator accesibilă pe bit
0029    10   BIT_0:      DBIT   1         ;Defineşte bitul BIT_0
002A    11   BIT_1:      DBIT   VAL2      ;Defineşte funcție de VAL2 bitul BIT_1
0008    12   VAL2        SET    8H        ;Redefineşte variabila VAL2
0008    13   BIT_2       BIT    VAL2      ;Defineşte bitul BIT_2
        14  XSEG   AT 80H
;Defineşte zona memorie externe utilizator începând de la adresa 80H
0080    15            STACK1   DATA  80H      ;Defineşte pointer-el STACK1
        16            CSEG        ;Defineşte începutul zonei "memorie de program"
0000    17            START   CODE    00H     ;Defineşte tipul memoriei
0000 020003    18        JMP     MAIN             ;Instrucțiune de salt
0003    19            ISR0 CODE    START+3    ;Defineşte adresă coresp. ISR0
000B   20             ISR1 CODE    START+0BH ; Defineşte adresă coresp. ISR0
0003    21            MAIN:       ;Adresă de salt (definită de eticheta MAIN)
0003 858081    22        MOV    SP,STACK1    ;Inițializare cu STACK1 a SP
         23     END        ;Sfârşit program sursă

SYMBOL TABLE LISTING
------ ----- -------
N A M E        T Y P E      V A L U E  ATTRIBUTES
BITI . . .         D ADDR        0021H   A
BIT_0. . .         B ADDR        0025H.1 A
BIT_1. . .         B ADDR        0025H.2 A
BIT_2. . .         B ADDR        0021H.0 A
COUNTER. .        REG     R0
ISR0 . . .         C ADDR        0003H   A
ISR1 . . .         C ADDR        000BH   A
MAIN . . .         C ADDR        0003H   A
REG23. . .         D ADDR        0010H   A
SP . . . .         D ADDR        0081H   A
STACK1 . .         D ADDR        0080H   A
START. . .         C ADDR        0000H   A
VAL. . . .         N NUMB        0020H   A
VAL1 . . .         N NUMB        0010H   A
VAL2 . . .         N NUMB        0008H   A
```

Alături de dispozițiile mai sus menționate, pentru uşurarea proiectării programelor sunt admise o serie de macro-instrucțiuni, specifice asamblorului, care asigură asamblarea condiționată a fişierelor sursă de program. Dintre avantajele pe care un astfel de stil în programare le are, amintim:

- Reduce substanțial frecvența erorilor acolo unde aceleaşi secvențe de instrucțiuni sunt utilizate.
- Scopul simbolurilor utilizate în MACRO-uri este limitat doar la macro-ul respectiv
- Este util în special pentru crearea unor tabele de coduri simple.

Iată aceste directive:

```
Nume_macro        MACRO      [listă_parametrii_formali][30]
                  LOCAL      Nume_simbol[31] [,Nume_Simbol1][…]
                  REPT       Număr_repetiții[32]

[Etichetă:]       IRP    parametru_formal <listă>
[Etichetă:]       IRPC   parametru_formal,parameru_actual
                  EXITM  comandă terminarea buclei macro curente
```

Exemple:

Am specificat deasupra liniei de definiție elementele acesteia:

```
Nume definiție MACRO        Cuvânt cheie          Parametrul MACRO
1   CLR_MEM                 MACRO                 DIMENS
2       MOV         R0,#DIMENS   ;Încarcă în R0 valoarea DIMENS
3   LOOP:                        ;Eticheta de definire a buclei
4       MOV         @R0,#00H     ;Scrie la adresa memorată în R0:00H
5       DJNZ        RO,LOOP      ;Testează şi ciclează până R0=0
6       ENDM                     ;Sfârşit MACRO CLR_MEM
; Macro-ul realizează umplerea cu zero a memoriei interne de la adresa 000H şi
; până la adresa DIMENS
              7   REP_OK:               ;Defineşte o etichetă de recunoaştere
              8       REPT   2          ;Defineşte factorul de repetiție
              9       MOV    R0,#1      ;Încarcă în R0 valoarea 1
             10       MOV    A,@R0      ;Transferă de la adresa 1 informația în A
             11    ENDM                 ;Sfârşit sursă program generat repetitiv
0000 7801      12+1   MOV    R0,#1      ;Programul generat (4 instrucțiuni)
0002 E6        13+1   MOV    A,@R0
0003 7801      14+1   MOV    R0,#1
0005 E6        15+1   MOV    A,@R0
0006         16    IRP_EX:       ;Exemplu de generare succesiune instrucțiuni
             17       IRP    X,<2,3> ;Definire variabilă X
```

---

[30] Parametri formali sunt variabile utilizate drept parametri în cadrul MACRO-ului

[31] Variabilă/simbol utilizat în spațiul macro-ului

[32] Variabilă ce precizează de câte ori se inserează în codul sursă blocul de instrucțiuni cuprins între REPT şi ENDM

```
               18           MOV   R0,X    ;Definire instrucțiune ce include X
               19           INC   R0      ;Instrucțiune de incrementare;
               20     ENDM                ;Sfârşit macro definiție repetitivă variabilă
0006 A802         21+1 MOV   R0,2    ;Cod generat cu prima valoare X
0008 08           22+1 INC   R0
0009 A803         23+1 MOV   R0,3    ;Cod generat cu a doua valoare X
000B 08           24+1 INC   R0
000C           25     IRPC_EX:
;Macro definiție repetivită ce utilizează variabila X dând valorile 2,5 şi 9
               26           IRPC  X,<259>         ;Setare valori variabilă X
               27           MOV   @R0,X           ;Definire instrucțiune
               28     ENDM        Sfârşit macro definiție repetitivă
000C A602      29+1         MOV   @R0,2  ;Definire instrucțiune cu valoarea X(1)
000E A605      30+1         MOV   @R0,5;Definire instrucțiune cu valoarea X(2)
0010 A609      31+1         MOV   @R0,9;Definire instrucțiune cu valoarea X(3)
               32     END
```

```
SYMBOL TABLE LISTING
------ ----- -------
N A M E     T Y P E  V A L U E   ATTRIBUTES
IRPC_EX. .      C ADDR          000CH   A
IRP_EX . .      C ADDR          0006H   A
REP_OK . .      C ADDR          0000H   A
```

Următoarele caractere sunt considerate caractere speciale în cazul macro-asamblorului A51.EXE:

- & concatenează textul cu parametrii "dummy"
- este utilizat pentru a introduce delimitatorii în text cum ar fi : , şi spațiul (blank)
- ; delimitează de la el până la sfârşitul liniei zona ce nu va fi procesată de către asamblor
- ! este folosit pentru a indica asamblorului că următorul caracter special va fi considerat literal.
- NUL este cuvântul cheie utilizat ca expresie a "non valorii", căci orice alt caracter, inclusiv blank-ul ar fi interpretat de către asamblor în cadrul unor expresii logice cum ar fi IF.

Apelarea unei expresii MACRO impune o anumită sintaxă, şi anume:

```
[etichetă:]        Nume_Macro              [Parametri_actuali]
```

Ordinea parametrilor avută în vedere la definirea MACRO-ului trebuie respectată întotdeauna la apelarea sa. Ea este singurul element ce asigură recunoaşterea corectă a parametrilor în cadrul macro-ului.

Asamblorul admite caracterul $ plasat în prima coloană a unei linii drept caracter "primar" de control. El va fi urmat, nu neaparat imediat, de diverse cuvinte cheie, cum sunt:

$ XREF/NOXREF activează/inhibă lista de referință cu variabilele utilizate de program
$ TITLE defineşte titlul programului, care poate fi format dintr-un şir de caractere (maximum 60) ce urmează cuvântului cheie
$ MOD51/NOMOD51 Defineşte/redefineşte setul de registre. Implicit asamblorul execută asamblarea considerând că programul este scris pentru un microcontroller standard 8051, în celelalte situații va trebui să inhibăm opțiunea prin dispoziția:
$ NOMOD51, urmată de dispoziția $ INCLUDE
$ MACRO/NOMACRO activează/inhibă procesorul de MACRO-uri
$ DEBUG/NODEBUG include sau nu informațiile de depanare necesare debug-erului sau emulatorului
$ DATE (30,02,00) cuvântul cheie este urmat de un şir ce exprimă data (maximum 9 caractere)
$ OBJECT: nume_fişier indică numele fişierului ce va include rezultatul asamblării, respectiv fişierul cu codul obiect
$ REGISTERBANK/NOREGISTERBANK (Numărul/numerele bancurilor de registre utilizate)
$ INCLUDE (Nume_fişier_cu_definiții_registre) permite specificarea între paranteze a fişierului ce defineşte structura de registre specifică procesorului pentru care a fost scris programul
$ LIST/NOLIST indică liniile din cadrul fişierului sursă ce nu vor fi listate în fişierul listing rezultat în urma asamblării

Utilizând dispozițiile de asamblare condiționată putem scrie compact şi incluzând toate variantele de implementare programe. În acest scop asamblorul pune la dispoziția programatorului următoarele dispoziții de asamblare condiționată.
$SET/RESET (<var1>[,<var2>], …) sau (<var>=<valoare_numerica>[,…])
$ IF expresie numerică ce va fi analizată dacă este sau nu adevărată, în caz afirmativ va fi asamblat programul ce urmează dispoziției până la întâlnirea dispoziției $ ENDIF

Alte două opțiuni pot completa dispoziția IF cu alternative la aceasta. Un exemplu va fi edificator:

```
LOC  OBJ          LINE     SOURCE
                     1     $DEBUG
;Dispoziție de generare a simbolorilor pentru depanarea programului
                     2     $SET    (A=1)
;Defineşte variabila A pentru comanda macro assembler-ului. Variabilei trebuie să
;i se atribuie o valoare atunci când este invocată operația de asamblare a codului
;sursă
                     3     $IF A=1          ;Testare variabilă A, Dacă este adevărată
;condiția, se execută asamblarea instrucțiunii următoare
```

```
0000 E599          4        MOV    A,SBUF;Instrucţiune de asamblat când A=1
                   5   $ELSEIF B=2
                     MOV    R0,SBUF
                   $ELSE              ;Alternativă la bifurcaţia IF
                     MOV    R1,SBUF        ;Instrucţiune alternativ asamblată
                   $ENDIF             ;Sfârşit bifurcaţie IF…. ELSE….
                   10   $RESET     (A)
;Eliberare memorie corespunzătoare variabilei A
                   11   $SET   (B=2)       ;Setare valoare pentru variabila B
                   12   $IF A=1
                     MOV    A,SBUF
                   $ELSEIF B=2
;Dispoziţie macroasamblor de testare iterativă tip IF. Este utilizată pentru
;simularea instrucţiunilor de tip CASE .. ON … ELSE
0002 A899          15        MOV    R0,SBUF
                   16   $ELSE
                     MOV    R1,SBUF
                   $ENDIF
                   19   $RESET  (B)
                   20   $IF A=1
                     MOV    A,SBUF
                   $ELSEIF B=2
                     MOV    R0,SBUF
                   $ELSE
0004 A999          25        MOV    R1,SBUF
                   26   $ENDIF
                   27   END
```

În încheiere prezentăm sintaxa liniei de comandă DOS ce invocă execuţia programului asamblor:

A51 Nume_fişier_sursă.a51 [PRINT:CO:]

Utilizarea link-editorului se face tot din mediul DOS , linia de comandă având structura:

L51 listă_fişiere_de_intrare [TO nume_fişier_ieşire] [listă_controale]<CR,LF>

Sau:

L51 fişier_de_comenzi <CR,LF>

Lista include nume de fişiere ce sunt separate prin caracterul ","

Numele fişierului de ieşire trebuie precizat atunci când dorim ca fişierul cu extensia logică ".M51" să fie specificat de către utilizator. În caz contrar, acest fişier va lua numele primului fişier de lista fişierelor de intrare.

Fişierul de comenzi poate conţine în format ASCII aceiaşi parametri ca şi cei din linia de comandă. Informaţii suplimentare găsiţi în cadrul CD-ului.

Un program deosebit de important este manager-ul de biblioteci LIB51.

Acest program asigură crearea unei biblioteci de funcţii şi actualizarea acesteia. Linia de comandă cu care este invocat programul are structura:

LIB51 [<comanda>]<CRLF>

Comenzile admise sunt:

Comanda	Funcţiunea
ADD	Adaugă un nou modul de program în bibliotecă
CREATE	Generează o bibliotecă de funcţii (iniţial goală)
DELETE	Şterge din bibliotecă un modul de program
EXIT	Revine în mediul DOS
HELP	Oferă scurte indicaţii asupra comenzilor
LIST	Afişează modulele şi simbolurile publice nume ale bibliotecilor

Comanda de adăugare (ADD) are sintaxa:

ADD nume_fişier [(nume_modul1,nume_modul2,…)] TO nume_bibliotecă <CRLF>

Comanda de ştergere (DELETE) are sintaxa

DELETE nume_bibliotecă (nume_modul) <CRLF>

Sistemele de dezvoltare sunt deosebit de diverse. Ele dispun în afară de microcontroller şi de alte elemente cum sunt porturile suplimentare, de obicei porturi de tip "registru" pe ieşiri şi "buffer"-e pe intrări. Având în vedere caracteristicile specifice microcontrollerelor selecţia porturilor este realizată în spaţiul dedicat memoriei externe a microcontroller-ului. (vezi nu este cazul pentru microcontroller-ele Atmel seriile 89Atxxxx (1051,2051,4051), ci variantele 89Atxx (51,52,55, etc) sau pentru microcontroller-ele firmelor Intel Philips, Dallas, etc. În figura 2.5 prezentăm un sistem de dezvoltare dotat cu un monitor Franklin ce are implementată o consolă via interfaţa serială. Sistemul dispune de 2 porturi de ieşire 1 port de intrare, precum şi un port de comunicaţie cu un afişor cu cristale lichide (LCD) şi un decodificator pentru circuite de intrare/ieşire cu 8 linii de selecţie.

Sistemul permite dezvoltarea unei aplicaţii care poate include toate elementele unui sistem dedicat. Evident dacă se doreşte, pot fi adăugate convertoare analog-digitale şi digital analoge, precum circuite de interfaţă diverse.

Avantajul principal al acestui sistem constă în aceea că el permite conectarea microcontroller-ului la procesul controlat ceea ce face ca utilizatorul să poată analiza, depana sau chiar reproiecta anumite programe chiar în contextul dat, având la dispoziţie toate semnalele de excitaţie reale.

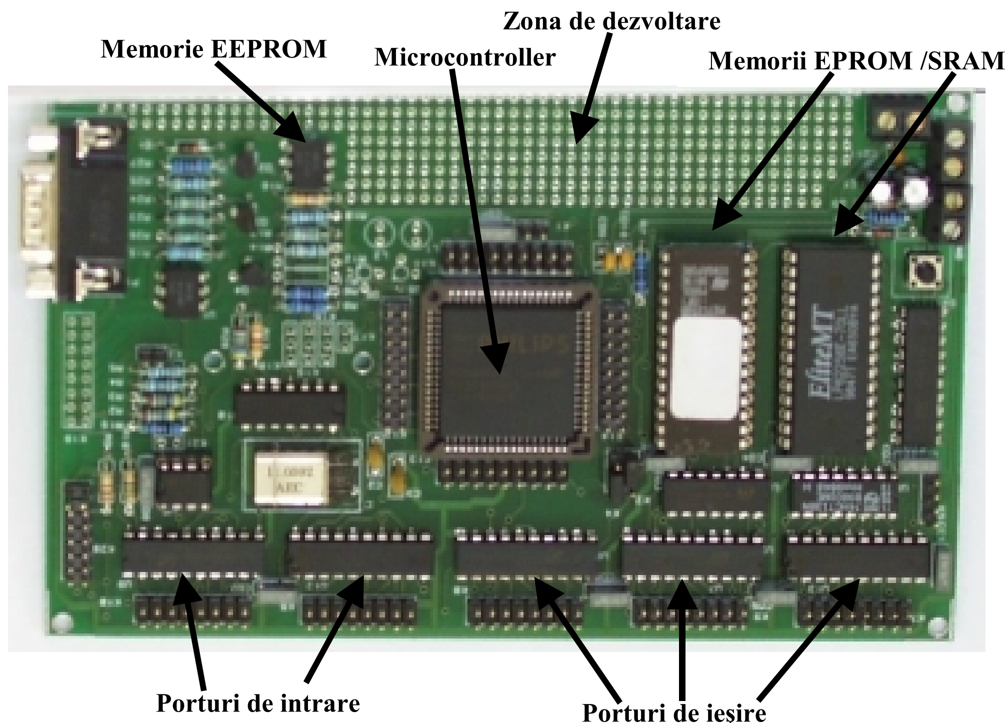


Figura 2. 5 Sistem de dezvoltare cu microcontroller PCB80C552

Programul monitor "mt.exe"funcționează împreună cu monitorul Franklin şi prezintă câteva comenzi:

A [adresă] <CR,LF> Asamblează cod 8051 de la adresa specificată
BD [ALL] < număr_break_point> Invalidează toate break point-urile sau cel specificat prin număr ce reprezintă chiar indexul acestuia
BE [ALL] < număr_break_point> Validează break point-urile.
BK [ALL] <număr_break_point> Şterge toate break point-urile sau doar cel specificat
BL Listează break point-urile setate
BS <adresă>Setează (stabileşte) un break point la adresa precizată
DB (*Display Bits*) <adresă_start> <adresă_stop> Afişează zonă memorie accesibilă pe bit între adresele de start şi stop
DD (*Display Data*) <adresă_start> <adresă_stop> Afişează zonă memorie accesibilă pe octet între adresele de start şi stop aflată în memoria internă a MC
DI (*Display Indirect Data*) <adresă_start> <adresă_stop> Afişează zonă memorie accesibilă pe octet indirect între adresele de start şi stop
DX (*Display Extended Data*) <adresă_start> <adresă_stop> Afişează zonă memorie accesibilă pe octet plasată în afara MC, între adresele de start şi stop

DC (*Display Code*) <adresă_start> <adresă_stop> Afişează zonă memorie accesibilă pe octet în memoria de program şi plasată între adresele de start şi stop
FILLB/FILLD/FILLI/FILLX/FILLC <adresă_start> <adresă_stop> <valoare> Directivă de umplere zonă de memorie, cu aceleaşi semnificaţii ca şi la Dispoziţia Display pentru diversele zone de memorie
G <adresă_start> <adresă_break> Dispoziţie de execuţie program între adresa de start şi cea de break
? Dispoziţie de help "on-line"
H<număr1><număr2> Dispoziţie de calcul în hexazecimal valoare sumă/diferenţă număr1, număr2
T <adresă_start> execută pas cu pas instrucţiunile programului cu afişarea tuturor registrelor microcontroller-ului
X[<nume_registru>] afişează toţi/registrul specificat
F1 tastă specială ce permite ieşirea din program
F2 tastă specială care permite încărcarea programelor în format Intel-HEX via interfaţa serială (consola)
F3 tastă specială ce permite salvarea în cadrul unui fişier al cărui nume este specificat de către operator a caracterelor recepţionate via interfaţa serială (consola)

2.1.4 Instrumente software de proiectare: MC PIC

Firma Microchip care produce microcontroller-ele PIC a adus pe piaţă produsul MPLAB drept mediu integrat de dezvoltare/simulare (IDE *Integrated Development Environment*) pentru procesoarele sale precum şi Starter-Kit-ul corespunzător fiecărui microcontroller. Mediul MPLAB cuprinde următoarele componente:

- Editorul fişierelor sursă
- Asamblorul
- Compilatorul C
- Linkeditorul
- Editorul de stimuli
- StarterPIC sistemul de dezvoltare minimal pentru dezvoltarea aplicaţiilor
- Simulatorul/Emulatorul microcontroller-elor PIC pentru diversele familii
- Programatorul microcontrollerelor ce dispun de memorie EPROM, EEPROM sau Flash.

Asamblorul este oferit atât integrat în mediu cât şi separat atât versiunea DOS cât şi versiunea Windows pe 16 biţi.

Mediul integrat dispune de ferestre ce permit atât editarea programului sursă, vizualizarea fişierului list rezultat în urma asamblării, fereasta ce vizualizează memoria internă de program şi de date a microcontroller-ului,

fereastra cu registrele acestuia (zona SFR) şi fereastra "trace", ce urmăreşte evoluția programului şi cea corespunzătoare memoriei EEPROM de care microcontroller-ele din familia PIC dispun.

Meniurile mediului integrat sunt în număr de 3 şi anume:cel corespunzător funcțiunilor de editare şi salvare fişiere şi proiecte, cel corespunzător procesării fişierelor sursă, respectiv asamblării sau compilării, al link-editării fişeirelor obiect şi cel corespunzător testării/simulării programelor scrise, ce permite execuția pas cu pas, admite editarea şi administrarea de stimuli sistemului precum şi în final, programarea microcontroller-ului respectiv.

În ceea ce însemnă dispozițiile de asamblorului MPASM, apar mici deosebiri față de asamblorul A51.EXE. În continuare vom prezenta câteva dintre aceste deosebiri, o listă completă de referință o puteți găsi pe CD.

Sistemul de numerație	**Sintaxa**	**Exemple**
Zecimal	D'<digiți>, .<digiți>	D'123', .255
Hexazecimal	H'<digiți>', 0x< digiți >	H'EF' , 0xE32A
Octal	O'< digiți >'	O'1234'
Binar	B'< digiți >'	B'11010011'
Caracter (ASCII)	'<caracter>', A'<caracter>'	'V', A'u'

Etichetele sunt exprimate prin şiruri de caractere şi ele identifică punctele de salt sau cele la care sunt plasate rutinele scrise.

Directiva	**Descriere**	**Sintaxa**
CONSTANT	Defineşte un simbol drept constantă în program	Constant<etichetă>=[<expresie>, <etichetă>=<expresie>,..]
#DEFINE	Defineşte text de substituție	#define <nume> [<arg>,<arg>,..]
EQU	Defineşte o constantă	<etichetă> equ <expresie>
#INCLUDE	Include un fişier sursă	include <nume_fişier>
PROCESSOR	Defineşte tipul procesorului	Processor <tip_procesor>
SET	Asignează o valoare unei variabile (poate fi apoi redefinită)	<etichetă> set <expresie>
#UNDEFINE	Renunță la definiția anterior precizată prin #DEFINE	#undefine <etichetă>
VARIABLE	Declară un simbol drept variabilă	variable <etichetă>[=expresie]
IF	Defineşte o macro-instrucțiune de decizie	if <expresie>
WHILE	Realizează buclarea atâta timp cât condiția testată la început este adevărată	while <>expresie

Directiva	Descriere	Sintaxa
BANKISEL	Selectează un banc de memorie internă pentru acces indirect	bankisel <etichetă>
RES	Rezervă spaţiu de memorie	res <număr_de_locații>
FILL	Umple o zonă de memorie cu o anume informaţie	fill <expresie>,<număr_locații>
DA	Împachetează în memoria de program un şir	<etichetă> da <expresie>[,<expresie>,…]
DATA	Crează o valoare numerică sau una text	data <expresie>[,<expresie>..], data <text>[,<text>,..]
DB	Declară un byte	db <expresie>[,<expresie1>,..]
DE	Declară o dată în EEPROM	de <expresie>[,<expresie1>,..]
DT	Defineşte o tabelă	dt <expresie>[,<expresie1>,..]
DW	Declară o dată un cuvânt	dw <expresie>[,<expresie1>,..]
MACRO	Defineşte un MACRO	<etichetă> macro [<argument>, <argument1>,..]
EXITM	Ieşire dintr-unMACRO	Exitm
GLOBAL	Exportă definiția unei etichete	global <etichetă>
__MAXRAM	Specifică adresa maximă pentru memoria RAM	__maxram <expresie>

Opțiunile de link-editorului MPLINK atunci când este invocat sunt:

/o nume_fişier	specifică fişierul de ieşire
/m nume_fişier	specifică generarea fişierului "hartă" nume_fişier
/l pathlist	specifică (adună)calea la fişierul bibliotecă ce este link-at
/k pathlist	specifică (adună) calea la fişierul script utilizat de linker
/n length	specifică numărul de linii llistate pe pagină
/h , /?	afişează fişierul de ajutor corespunzător link-erului
/a hexformat	specifică formatul HEX pentru fişierele de ieşire
/q	iese din link-are
/d	evită crearea fişierului listing absolut

3

MICROCONTROLLERE MOTOROLA (MC cu arhitectură CISC)

Motorola a dezvoltat câteva tipuri de unități centrale, la care s-au ataşat o multitudine de interfețe, obținând astfel foarte multe tipuri de circuite, compatibile din punct de vedere software. Motorola a creat astfel posibilitatea producerii a nenumărate variante de MC, care să acopere cât mai multe din aplicațiile utilizatorului, numind aceste structuri CSIC (Customer Specified Integrated Circuit). Spre deosebire de familia 8051 unde pe nucleul creat de INTEL, firme constructoare au dezvoltat propriile MC compatibile, Motorola a creat o familie de MC, obținând astfel o unitate în diversitate.

Schema bloc simplificată a unui MC din familia Motorola este dată în figura 3.1.

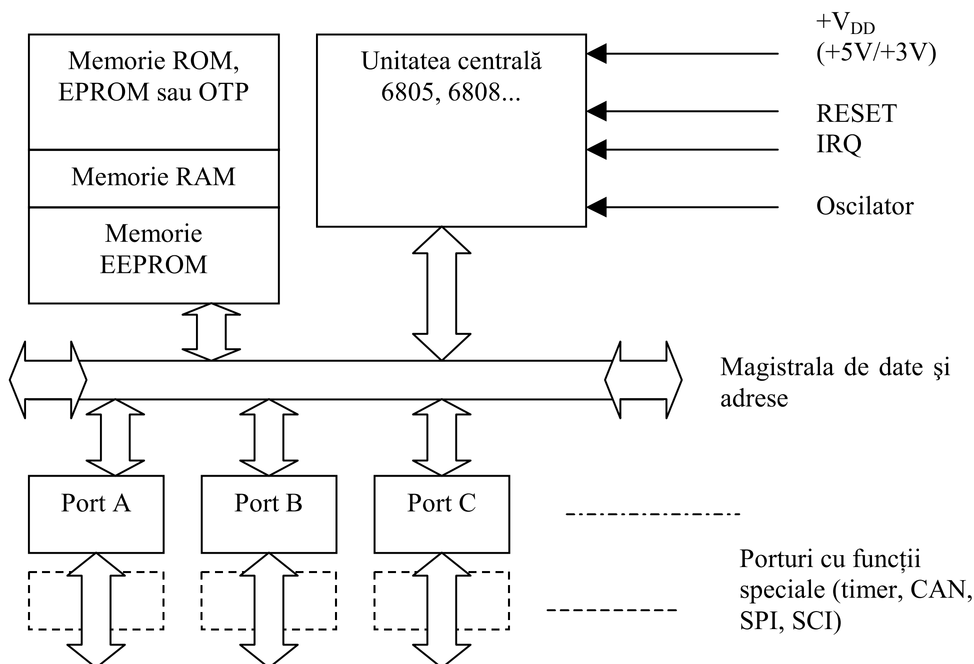


Figura 3.1 Schema bloc a unui microcontroller Motorola

Porturile I/O pot fi de uz general, dar liniile porturilor pot fi utilizate şi de interfeţele speciale. În continuare vor fi prezentate blocurile din structura MC Motorola 6805.

3.1 MC M68HC05

3.1.1 Memoria internă

MC din familia Motorola, ca şi toate circuitele MC, sunt echipate cu memorie RAM, ROM şi EPROM (OTP). Fiecare din memorii are o capacitate specifică fiecărui tip particular de circuit. Memoria poate fi formată din 176 – 304 octeţi de RAM, 240 octeţi de ROM şi 7600 – 7744 octeţi de memorie programabilă (EPROM sau OTP.

Memoria RAM este folosită pentru stocarea temporară a datelor. În modul de lucru bootstrap (specific MC Motorola), memoria RAM poate fi încărcată cu informaţie de tip program prin portul SCI, urmând ca programul să fie executat din RAM. Memoria ROM este folosită pentru a stoca programe. Modulul ROM conţine şi programul care coordonează încărcarea RAM prin portul SCI. Memoria EPROM (OTP), ca şi memoria ROM, conţine programe şi variabile specifice aplicaţiei cu diferenţa că acestea sunt înscrise de utilizator, nu de fabricant.

În acelaşi spaţiu de adresare se află memoria de diferite tipuri, registre I/O şi registre de control şi stare, privite ca locaţii de memorie. Pentru a şti adresa la care se află fiecare element se utilizează o reprezentare numită harta memoriei. Harta memoriei pentru un membru al familiei 6805 (MC68HC705J1A) este dată în tabelul 3.1. Pentru a putea lucra cu un MC particular trebuie studiată alocarea (maparea) memoriei din catalog pentru acel MC, tabelul 3.1 este doar orientativ.

Tabelul 3.1

Maparea memoriei la MC68HC05J1A

Conţinut	Adresa (H)
Date port A	0000
Date port B	0001
Nefolosit	0002-0003
DDR pentru port A	0004
DDR pentru port B	0005

Conținut	Adresa (H)
Nefolosit	0006-0007
Stare şi control pentru timer	0008
Registru numărător pentru timer	0009
Stare şi control a întreruperilor	000A
Nefolosit	000B-0011
Nefolosit	0012-0017
Registru de programare a EPROM	0018
Nefolosit	0019-001F
Nefolosit	0020-00BF
RAM utilizator sau stivă (64 octeți)	00C0-00FF
Nefolosit (512 octeți)	0100-02FF
EPROM (1232 octeți)	0300-07CF
Nefolosit (30 octeți)	07D0-07ED
Test ROM (2 octeți)	07EE-07EF
Registru COP	07F0
Registru de măşti	07F1
Rezervați	07F2-07F7
Vector pentru timer (octet semnificativ)	07F8
Vector pentru timer	07F9
Vector pentru IRQ (octet semnificativ)	07FA
Vector pentru IRQ	07FB
Vector pentru SWI (octet semnificativ)	07FC
Vector pentru RESET (octet semnificativ)	07FE
Vector pentru RESET	07FF

3.1.2 Unitatea centrală

Unitatea centrală realizează prelucrarea datelor pe 8 biți la o frecvență internă de 2MHz (tact extern de 4MHz), are o magistrală de adrese de 11-14 biți şi conține registre şi unitatea aritmetică şi logică, figura 3.2. În general, la MC Motorola, bus-ul de adrese şi de date nu este disponibil în exterior.

Unitatea centrală are o arhitectură cu bus unic pentru date şi instrucțiuni (von Neumann) şi conține următoarele registre:

Registrul Acumulator (*Accumulator* - *A*) este un registru pe 8 biți de uz general. El nu este afectat de Reset.

Registrul index (*Index Register* - *X*) este un registru pe 8 biți folosit la adresările indexate. Nu este afectat de Reset.

Registrul indicator de stivă (*Stack Pointer* - *SP*) este un registru de 13 biți care conține adresa următoarei locații libere în stivă. După Reset indicatorul de stivă devine 00FFh.

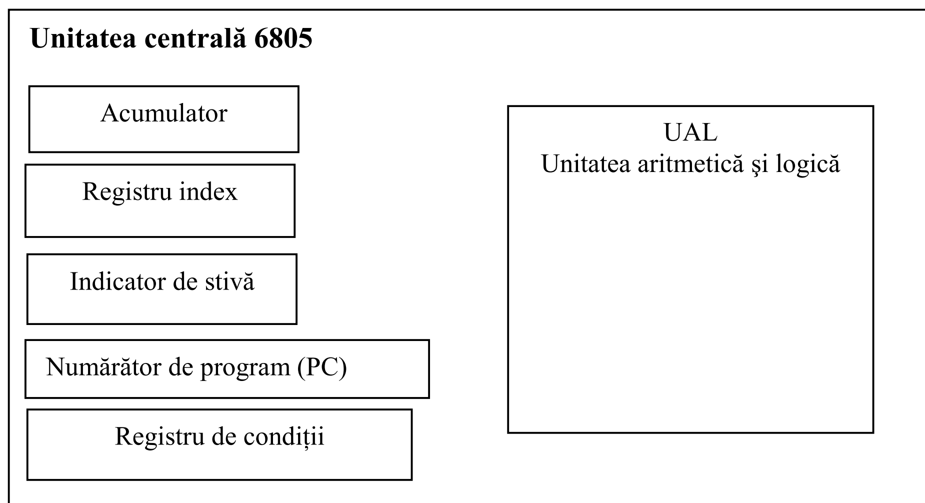


Figura 3.2. Unitatea centrală 6805

Registrul numărător de program (*Program Counter* - *PC*) este un registru de 13 biți care conține adresa următoarei instrucțiuni de executat. După execuția instrucțiunii registrul este incrementat. Un salt sau o întrerupere determină încărcarea PC cu altă valoare decât adresa următoarei locații. După Reset registrul PC este încărcat cu conținutul locațiilor 3FFEh şi 3FFFh (la MC68HC705J1A).
Registrul de condiții (*Condition Code Register* - *CCR*) este un registru de 8 biți (din care sunt folosiți 5) cu următoarea semnificație:

- Bitul 0 -LSB, (*Carry/Borrow Flag*) poziționat dacă o adunare produce un transport sau dacă la o scădere este nevoie de împrumut.
- Bitul 1 (*Zero Flag*), poziționat dacă rezultatul unei operații este zero.
- Bitul 2 (*Negative Flag*), poziționat dacă rezultatul unei operații este negativ (bitul 7 este 1).
- Bitul 3 (*Interrupt Mask Bit*) când este 1 sunt invalidate întreruperile. Dacă apare o întrerupere când acest bit este 1, ea este memorată până bitul devine 0; atunci se salvează registrele UC în stivă. După Reset bitul este setat în 1 şi poate fi poziționat în 0 cu instrucțiunea CLI.
- Bitul 4 (*Half-Carry*) este poziționat când apare un transport de la bitul 3 spre 4 în acumulator la o operație de adunare cu sau fără Carry. Este util la operații BCD.

Unitatea aritmetică şi logică realizează operațiile aritmetice şi logice definite de setul de instrucțiuni. Multe din instrucțiunile aritmetice se bazează pe algoritmul de la adunare. Astfel, o înmulțire este realizată printr-un şir de adunări şi deplasări succesive şi durează 11 perioade de clock.

Inițializarea unității centrale

Un Reset duce unitatea centrală într-o stare determinată (la o adresă cunoscută). Un Reset poate fi declanşat de o sursă internă sau de o sursă externă astfel:

- extern, prin pinul de RESET, activ pe 0;
- extern, Power On Reset (POR), la pornirea sursei de alimentare. Circuitul POR asigură o întârziere de 4064 tacte de la momentul în care oscilatorul devine activ şi dacă semnalul RESET extern este inactiv, procesorul începe să lucreze;
- intern, de către ceasul de gardă, Computer Operating Properly (COP).
- încercarea de a executa o instrucțiune de la o adresă ilegală. Dacă unitatea centrală încearcă să execute o instrucțiune care nu este în EPROM sau RAM se generează un RESET pentru a proteja MC de scrieri sau citiri din zone necontrolate.

3.1.3 Moduri de adresare

Unitatea centrală foloseşte 8 moduri de adresare pentru o cât mai mare flexibilitate în utilizare:

1. **adresare inerentă -** instrucțiunile nu au operand, cum este întoarcerea din întrerupere (RTI) sau STOP. Instrucțiunile inerente au lungimea de un octet.
2. **adresare imediată -** instrucțiunile au un operand, care este o valoare imediat utilizabilă într-o operație cu acumulatorul sau cu registrul index. Instrucțiunile au doi octeți, primul fiind codul, iar al doilea valoarea imediată.
3. **adresare directă -** instrucțiunile pot avea acces la primele 256 de locații de memorie. Instrucțiunile sunt pe doi octeți, primul este codul iar al doilea este octetul cel mai puțin semnificativ al adresei operandului; octetul cel mai semnificativ este considerat implicit 00h.
4. **adresare extinsă -** instrucțiunile sunt pe trei octeți şi pot adresa orice locație de memorie. Primul octet este codul, al doilea este adresa (parte HIGH) iar al treilea este adresa (partea LOW).
5. **adresare indexată fără offset** - instrucțiunile au un octet şi pot avea acces la locații cu adresa variabilă din registrul index (care conține partea LOW a adresei). Pentru partea HIGH se foloseşte 00h, aşa încât aceesul este în zona 0000-00FFh.
6. **adresare indexată cu offset pe 8 biți -** instrucțiunile au doi octeți, fiind accesibile locații cu adresa variabilă în zona primelor 511 locații. Unitatea centrală adună octetul din instrucțiune la registrul index (fără semn) şi se obține adresa efectivă a operandului. Un exemplu de utilizare al acestui mod de adresare este selectarea unui element k dintr-un tabel de n elemente. Valoarea

k este în registrul index, iar adresa începutului tabelului este al doilea octet al instrucțiunii.

7. **adresare indexată cu offset pe 16 biți -** instrucțiunile au trei octeți, fiind accesibile toate locațiile. UC adună fără semn registrul index la cei doi octeți din instrucțiune şi se obține adresa efectivă a operandului (primul octet după codul instrucțiunii este octetul HIGH).
8. **adresare relativă –** este folosită în instrucțiunile de salt. La un salt relativ, se adună (cu semn) octetul care urmează după codul instrucțiunii la conținutul registrului Program Counter. Se pot face salturi relative în gama 127 de octeți înainte sau înapoi.

3.1.4 Setul de instrucțiuni

Unitatea centrală MC68HC05 are 61 de instrucțiuni (mai mult de 200 de coduri). Câteva dintre instrucțiunile mai importante, clasificate după tipul operației, sunt enumerate în continuare.

Operații registru- memorie

ADD, ADC - adună conținutul unei locații de memorie la acumulator (fără sau cu transport)
AND - şi între memorie cu acumulator
CMP - compară memorie cu acumulator
EOR - sau exclusiv memorie cu acumulator
CPX - compară registru index cu memoria
LDA, LDX - încarcă acumulator (registru index) cu un octet din memorie
ORA - sau logic între acumulator şi memorie
SBC, SUB - scădere cu sau fără împrumut
STA, STX - salvare acumulator (registru index) în memorie
MUL - înmulțire

Operații de tip citeşte- modifică- scrie

ASL,ASR, LSL, LSR - deplasare aritmetică (logică) stânga (dreapta)
BSET, BCLR - setare sau resetare bit
CLR - resetare registru
COM - complement față de 1
NEG - complement față de 2 (negare)
ROL, ROR - rotație stânga (dreapta) prin Carry
TST- - test pentru zero sau negativ

Operații de salt

BCC, BCS - dacă bitul Carry este 0 sau 1
BEQ, BNE - dacă este egal (sau nu)
BHCC, BHCS - dacă bitul Half Carry este 0 sau 1
BHI, BHS - dacă este mai mare ori mai mare sau egal
BLO, BLS - dacă este mai mic ori mai mic sau egal

BIH, BIL - dacă linia de întrerupere externă este 1 sau 0
BMI, BPL - dacă este negativ (sau pozitiv)
BMC, BMS - dacă masca de întrerupere este 0 sau 1
BRA, JMP - salt necondiționat
BSR, JSR - salt la subrutină

Manipulare la nivel de bit

BCLR, BSET - înscrierea unui bit cu 0 (sau cu 1)
BRCLR, BRSET - salt dacă un bit este 0 (sau cu 1)

Operații de control

CLC, SEC - înscriere cu 0 (cu1) a bitului de Carry
CLI, SEI – înscriere cu 0 (cu 1) a măştii pentru întrerupere
NOP - nu se execută nici o operație
RTI, RTS - întoarcere din întrerupere (subrutină)
STOP, WAIT - se opreşte oscilatorul CPU şi se aşteaptă o întrerupere externă (se validează întreruperile)
SWI - întrerupere software
TAX, TXA - transferă acumulatorul în registrul index (sau invers)

3.1.5 Sistemul de întreruperi

Unitatea centrală poate fi întreruptă din programul curent de următoarele surse:

- un 0 logic aplicat din exterior pinului nIRQ;
- un 1 logic aplicat din exterior la pinii PA3-PA0 ai portului I/O PA, dacă aceste întreruperi sunt validate;
- o întrerupere de la timer-ul sistemului (TOF Timer Overflow sau RTIF Real Time Interrupt), dacă întreruperea este validată;
- instrucțiunea de întrerupere software SWI.

Dacă o întrerupere vine în timp ce UC execută o instrucțiune, instrucțiunea în curs este terminată şi apoi se consideră întreruperea. Întreruperile pot fi invalidate global în registrul condițiilor de program (CCR) şi individual pentru fiecare sursă în parte. Un Reset inhibă toate întreruperile pentru ca procesul de inițializare să nu poată fi întrerupt.

La apariția unei întreruperi, unitatea centrală termină instrucțiunea în curs, apoi salvează în stivă registrele UC, invalidează întreruperile pentru ca o nouă întrerupere să nu deranjeze servirea întreruperii curente. Cererile de întrerupere sunt memorate şi servite după servirea întreruperii curente. În PC se transferă vectorul de întrerupere şi se execută rutina de servire a întreruperii. Rutina se termină cu instrucțiunea RTI care restaurează registrele UC din stivă. Vectorii de întrerupere pentru fiecare dintre sursele enumerate mai sus sunt stocați în memorie la adresele:

- timer 07F8h şi 07F9h;
- nIRQ sau PA 07FAh şi 07FBh;
- instrucţiunea SWI 07FCh şi 07FDh;
- Reset (POR, RESET din exterior, COP sau adresă ilegală) 07Feh şi 07FFh.

Adresele vectorilor pentru fiecare tip de MC se găsesc în harta memoriei furnizată de producător.

Unele MC din familie au şi alte interfeţe care pot şi ele să solicite întreruperi, aşa cum poate cere circuitul timer.

Până la servirea întreruperii trece un anumit interval de timp (tacte), timp necesar pentru ca UC să termine instrucţiunea curentă şi să salveze în stivă registrele. Acest timp este numit *Interrupt Latency* şi poate fi oricât de lung dacă întreruperile nu sunt validate. Cea mai lungă instrucţiune este MUL, care durează 11 tacte, iar unitatea centrală are nevoie de 9 tacte pentru a salva registrele în stivă, deci cel mai lung timp de aşteptare pentru servirea unei întreruperi este de 20 de tacte. Acest timp trebuie să fie luat în calcul în momentul realizării unei aplicaţii în timp real. Dacă survine o a doua întrerupere, timpul de aşteptare pentru servirea ei poate fi prea lung. O soluţie ar fi validarea întreruperilor în timpul servirii primei întreruperi. În acest caz trebuie avut grijă ca stiva să fie destul de mare pentru a permite salvarea a două seturi de registre.

3.1.6 Interfeţe şi periferice on-chip

Blocurile funcţionale integrate în circuitul MC sunt de o deosebită importanţă în implementarea unei aplicaţii. Este în egală măsură important să fie cunoscute şi bine stăpânite atât capabilităţile unităţii centrale cât şi ale interfeţelor cu care este echipat MC – această cunoaştere conduce la o exploatare performantă a resurselor.

a. Porturi paralele I/O

Porturile paralele I/O sunt forma cea mai simplă a interfeţelor. MC68HC705C8 dispune de 31 de linii I/O digitale de uz general grupate în patru porturi. Porturile A, B, şi C sunt porturi de câte opt biţi fiecare; sensul transferului pe fiecare linie (intrare sau ieşire) poate fi stabilit prin program. Fiecărei linii îi este asociat un bit într-un registru de direcţie (DDR *Data Direction Register*). Organizarea porturilor de opt biţi (A,B şi C) este identică.

Portul D dispune de şapte linii (fără PD6); acestea pot fi folosite doar ca linii de intrare. O particularitate a acestui port este faptul că pinii corespunzători sunt folosiţi fie de portul D, fie de porturile seriale SPI şi SCI, în funcţie de starea de activare înscrisă în registrele de comandă ale porturilor seriale SPI şi SCI.

Liniile I/O pot fi citite sau scrise de unitatea centrală cu instrucţiuni specifice. Liniile pot fi apelate grupat, la nivel de port sau individual, la nivel de

bit. Pentru a economisi timpul unității centrale, interfețele pot lucra cu unitatea centrală prin intermediul întreruperilor.

b. Portul serial asincron SCI (*Serial Communications Interface*)

SCI este un transmițător receptor asincron universal (UART) cu posibilitatea lucrului full-duplex. Pentru un transfer bidirecțional sunt suficienți doi pini. Cu circuite de translație de nivel RS232 se pot face transferuri la distanțe suficient de mari. Se poate programa una din 32 viteze de transmisie şi lungimea caracterului; se pot valida separat transmițătorul şi receptorul; se pot genera întreruperi în diferite situații; se poate detecta eroare de cadrare la recepție.

Formatul datelor este ca la orice transmisie asincronă (RS232 sau RS422). Linia este în stare MARK, iar transmisia unui caracter este semnalată de trecerea liniei în stare SPACE pe durata bitului de START. Urmează 8 sau 9 biți de date şi un bit de STOP.

Datele care se doresc a fi transmise sunt scrise în registrul de date al SCI (SCDR), apoi se validează transmisia prin poziționarea bitului TE (*Transmit Enable*) în registrul de control al SCI (SCCR2). După ce cuvântul a fost transmis, se poziționează bitul TDRE (*Transmit Data Register Empty*) în registrul de stare SCSR (*SCI Status Register*). Se indică astfel că poate fi transmis un nou cuvânt. Se pot transmite caractere speciale, cum ar fi BREAK, care ține linia în stare SPACE (se transmite un şir de 0) dacă se poziționează bitul SBK în SCCR2 sau un caracter care ține linia în stare MARK.

Golirea registrului de date semnalizată de TDRE sau de TC (*Transmission Complete*) din SCDR poate genera o întrerupere.

Datele sunt recepționate în SCDR, la recepția completă fiind poziționat bitul RDRF (*Receive Data Register Full*) în SCSR. Pentru eliminarea recepțiilor false fiecare bit de start este eşantionat şi citit de 16 ori; orice nepotrivire a eşantioanelor duce la respingerea acestui bit. Dacă nu se recepționează bitul de STOP se anunță o eroare de cadrare prin poziționarea bitului FE în SCR.

Întreruperile la recepție pot fi generate dacă bitul RDRF din SCSR indică recepția unui caracter, dacă este o eroare de viteză de recepție (prin recepția unui caracter înainte ca cel precedent să fi fost citit) sau dacă s-a recepționat un caracter special format numai din valori de 1.

Structura registrului de control SCCR1 este:

Bit 7 este al 9-lea bit recepționat dacă s-a definit o transmisie pe 9 biți

Bit 6 este al 9-lea bit de transmis dacă s-a definit o transmisie pe 9 biți

Bit 4 comandă lungimea caracterului, 8 sau 9 biți

Bit 3 este un bit de trezire care comandă ce fel de condiție trezeşte SCI

Structura registrului de control SCCR2 este:

Bit 7 - TIE *Transmit Interrupt Enable* - validează întreruperile cerute de TDRE

Bit 6 - TCIE *Transmit Complete Interrupt Enable* - validează întreruperile cerute de TC

Bit 5 - RIE *Receive Interrupt Enable* - validează întreruperile cerute de RDRF

Bit 4 - ILIE *Idle Interrupt Enable Bit* -validează întreruperile cerute de caracterul şir de 1

Bit 3 - TE *Transmit Enable* - validează transmisia

Bit 2 - RE *Receive Enable* - validează recepţia

Bit 1 - RWU *Receiver Wakeup Enable* - pune receptorul în stare de aşteptare

Bit 0 - SBK *Send Break* - trimite caracterul BREAK

Structura registrului de stare SCSR este:

Bit 7 - TDRE *Transmit Data Register Empty* - registru de transmisie gol

Bit 6 - TC *Transmission Complete* - transmisie completă

Bit 5 - RDRF *Receive Data Register Full* - registru de recepţie plin

Bit 4 - IDLE - s-a recepţionat un şir de 1

Bit 3 - OR *Receiver overrun* - eroare de viteză de recepţie

Bit 2 - NF *Receiver Noise* - s-au detectat perturbaţii în datele citite (prin eşantionarea bitului de START)

Bit 1 - FE *Framing Error* - eroare de cadrare.

Registrul ratei de transfer este descris în tabelul 3.2.

Tabelul 3.2

Conţinutul registrului ratei de transfer

Bit 5	Bit 4	Ceas pentru transfer	Bit 2	Bit 1	Bit 0	Rata de transfer
0	0	ceas intern	0	0	0	ceas de transfer
0	1	ceas intern /3	0	0	1	ceas de transfer/2
1	0	ceas intern/4	0	1	0	ceas de transfer/4
1	1	ceas intern/13	0	1	1	ceas de transfer/8
			1	0	0	ceas de transfer/16
			1	0	1	ceas de transfer/32
			1	1	0	ceas de transfer/64
			1	1	1	ceas de transfer/128

Rata de transfer se obţine printr-o dublă divizare, întîi se obţine un ceas de transfer prin divizarea ceasului intern, apoi se divizează ceasul de transfer. Se pot astfel obţine diferite rate de transfer, de exemplu 4800 Baud (4808) la un ceas de 2MHz prin divizarea întâi cu 13, apoi cu 1.

c. Portul serial SPI (Serial Peripheral Interface)

Cu acest port se poate realiza o comunicaţie sincronă simplă, folosită de regulă pentru a transfera date între circuite pe acelaşi montaj cu MC. Un transfer bidirecţional necesită 3 pini, unul dintre ei fiind alocat ceasului de transmisie generat de masterul SPI. Cu SPI se pot realiza transferuri şi între microcontroller-e. Transferurile pot fi full duplex.

Registrele care controlează transferul SPI sunt registrul de control (SPICR - *SPI Control Register*) şi registrul de stare (SPISR - *SPI Status Register*). Un transfer SPI poate fi iniţiat doar de un master. Master-ul scrie un octet în registrul

de transmisie SPI (SPDR - *SPI Data Register*) de unde datele merg într-un registru de deplasare în care sunt serializate şi de unde sunt transmise sincron cu ceasul de transmisie. Transmisia se termină după 8 tacte, când se poziţionează bitul SPIF. Înainte ca master-ul să trimită un nou octet trebuie să se reseteze bitul SPIF prin citirea registrul de stare SPSR. În slave datele intră în registrul de deplasare cu tactul de recepţie, acelaşi cu cel de transmisie. Când au intrat 8 biţi, caracterul este transferat în registrul de date SPDR. Pentru a se evita erorile de viteză (sau de suprascriere - *Overrun*) trebuie ca octetul din SPDR să fie citit înainte ca un alt octet să fie transmis din registrul de deplasare.

Poziţionând bitul MSTR din registrul SPCR în 1, MC lucrează în mod master. În acest mod pinii au următoarea semnificaţie:

- SCK (*Serial Clock*) este ieşire de tact pentru sincronizare;
- MOSI (*Master Output Slave Input*) este ieşirea serială;
- MISO (*Master Input Slave Output*) este intrarea serială;
- nSS (*Slave Select*) protejează MC dacă două circuite sunt master. Acest semnal activ dezactivează la celălalt port SPI modul master, şterge bitul MSTR şi poziţionează bitul de eroare (MODF *Mode Fault Flag*).

Cu bitul MSTR=0 se validează modul slave, în care pinii au semnificaţia:

- SCK (*Serial Clock*) este intrarea de tact pentru sincronizare de la master;
- MOSI (*Master Output Slave Input*) este intrarea serială;
- MISO (*Master Input Slave Output*) este ieşirea serială;
- nSS (*Slave Select*) validează SPI pentru modul slave.

În figura 3.3 este arătată o conexiune SPI în care sunt legate un circuit master şi mai multe circuite slave.

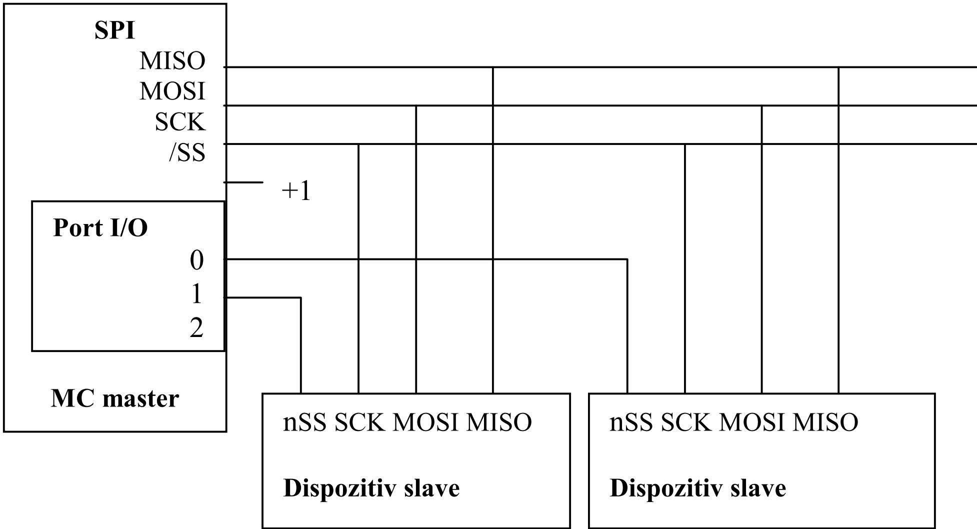


Figura 3.3 Conexiune SPI

Circuitele slave sunt validate pe rând cu semnale dintr-un port de ieşire auxiliar.

Pentru a putea adapta transmisiei seriale cât mai multe echipamente seriale, se poate programa faza şi polaritatea ceasului de sincronizare cu biții CPOL şi CPHA din SPCR.

În SPI pot să apară următoarele erori:

- mai multe MC master conectate (*Mode Fault Error*)
- scrierea în SPDR în timpul unei transmisii (coliziune), are ca urmare înscrierea bitului WCOL în SPSR;
- omiterea citirii SPDR înainte ca următorul octet să sosească (suprascriere).

O întrerupere poate fi generată în următoarele situații:

- registrul de date este gol sau plin (la transmisie respectiv la recepție) se semnalează cu SPIF care poate genera o întrerupere dacă întreruperea este validată cu SPIE;
- la apariția unei erori Mode Fault semnalată cu MODF, dacă întreruperea este validată cu SPIE.

d. Timer

Circuitele de timp care echipează MC Motorola sunt foarte variate. De la cel mai simplu (MC68HC05J) care poate genera 2 întreruperi periodice: una cu frecvența fixă şi una cu frecvența variabilă, şi până la cel mai complex (MC68332) care conține un temporizator cu propria lui unitate aritmetică şi logică proiectat special pentru controlul motoarelor cu ardere internă.

Un timer tipic (de la MC68HC705J1A), nu cel mai simplu dar nici cel mai complex, are schema bloc dată în figura 3.4.

Ceasul de intrare în timer este ceasul de magistrală (frecvența cristalului/2) care, după o divizare cu 4, constituie tact pentru un numărător de 8 biți. Valoarea acestui registru numărător poate fi citită de UC în oricare moment la locația 09h (TCR *Timer Counter Register*). UC nu poate să modifice valoarea acestui registru. Când numărătorul trece de la FFh la 00h este poziționat bitul TOF *(Timer Overflow Flag*) în registrul de stare al timer-ului (TCSR *Timer Control and Status Register*). Starea acestui bit poate fi citită de UC în oricare moment. Dacă bitul TOIE (*Timer Overflow Interrupt Enable*) din registrul de stare TCSR este 1, în momentul trecerii de la FFh la 00h se generează o întrerupere, numită întrerupere de depăşire (*Overflow Interrupt*).

Ieşirea numărătorului de 8 biți constituie tact pentru un alt numărător de 7 biți. Ieşirea de la oricare dintre cei mai semnificativi 4 biți ai acestui ultim numărător (selectată de unul din cei doi biți RT1 şi RT0 din registrul de control TCSR) poate fi folosită pentru generarea unei întreruperi, numită de timp real (*Real Time Interrupt*) care înscrie bitul RTIF (*Real Time Interrupt Flag*) din TCSR. Se generează o întrerupere dacă bitul RTIE (*Real Time Interrupt Enable*) este 1.

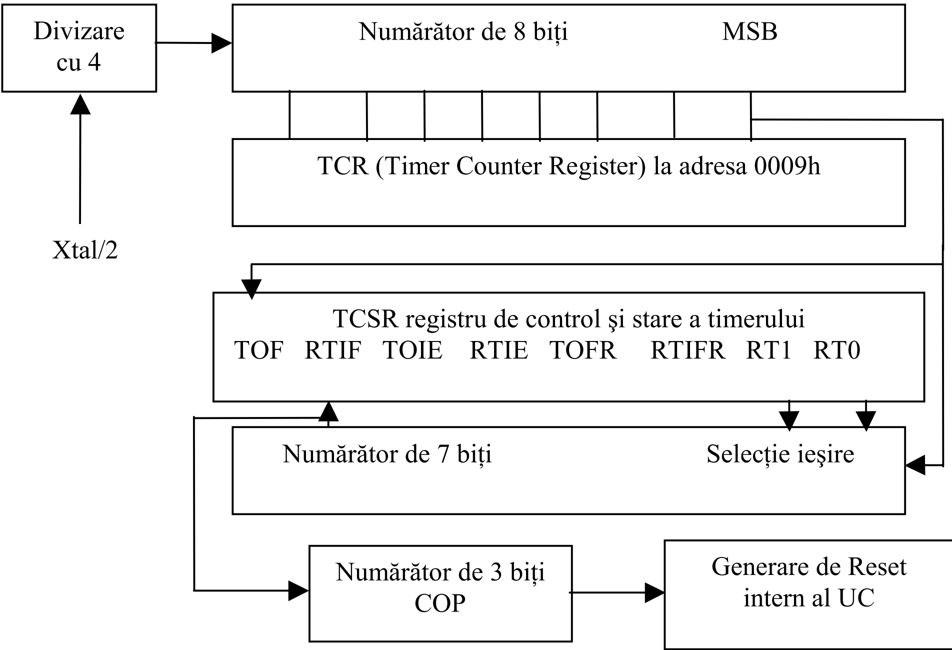


Figura 3.4 Timer 6805

Ultimul nivel al timer-ului este un numărător de 3 biți folosit pentru cesul de gardă (COP *Computer Operating Properly*). Dacă este validată verificarea COP, trebuie ca programatorul să reseteze COP înainte de expirarea perioadei programate pentru comanda unui RESET al UC. Perioada de timp în care COP trebuie resetat depinde de RT1 şi RT0. În tabelul 3.3 sunt date perioadele de întrerupere şi perioadele în care COP trebuie resetat. Perioadele corespund unui tact de 2MHz:

Tabelul 3.3

Perioade de întrerupere şi perioade de reset programabile cu un tact de 2 MHz.

RT1	RT0	Perioada de întrerupere	Perioada în care COP trebuie resetat
0	0	8.2ms	57.3ms
0	1	16.4ms	114.7ms
1	0	32.8ms	229.4ms
1	1	65.5ms	458.8ms

Un reset al COP se poate realiza prin scrierea bitului 0 de la adresa 07F0h (COPR *COP Register*).

MC din seria 68705 sunt echipate cu un timer care are posibilitatea de captură şi comparare. Captura poate înregistra momentul la care apare un eveniment extern (o tranziție pe pinul TCAP). În acest moment memorează conținutul registrelor timer-ului în registrele de captură. Memorarea registrelor timer-ului la tranziții de aceeaşi polaritate a TCAP poate determina perioada unui

semnal, iar la tranziţii de polaritate opusă poate determina lăţimea unui impuls. Polaritatea de declanşare este programabilă. Prin comparare se poate genera un semnal de ieşire când numărătorul timerului atinge o valoare selectată. La fiecare 4 tacte se compară valoarea numărătorului cu cea scrisă în registrul de comparare. Dacă este egalitate se generează un semnal TCMP.

e. Portul PWM

Modulaţia impulsurilor în lăţime (*Pulse Width Modulation*) are multe aplicaţii, mai ales în comanda motoarelor de curent continuu sau a surselor de alimentare. Din acest motiv unele MC includ în structura lor un modulator PWM ca interfaţă.

Circuitul MC68HC05D9 conţine 5 canale PWM de câte 6 biţi care sunt realizate cu ajutorul unui numărător de 6 biţi, un registru de control PWM şi 5 registre de date care formează 5 linii PWM ce sunt disponibile la pinii portului D (figura 3.5).

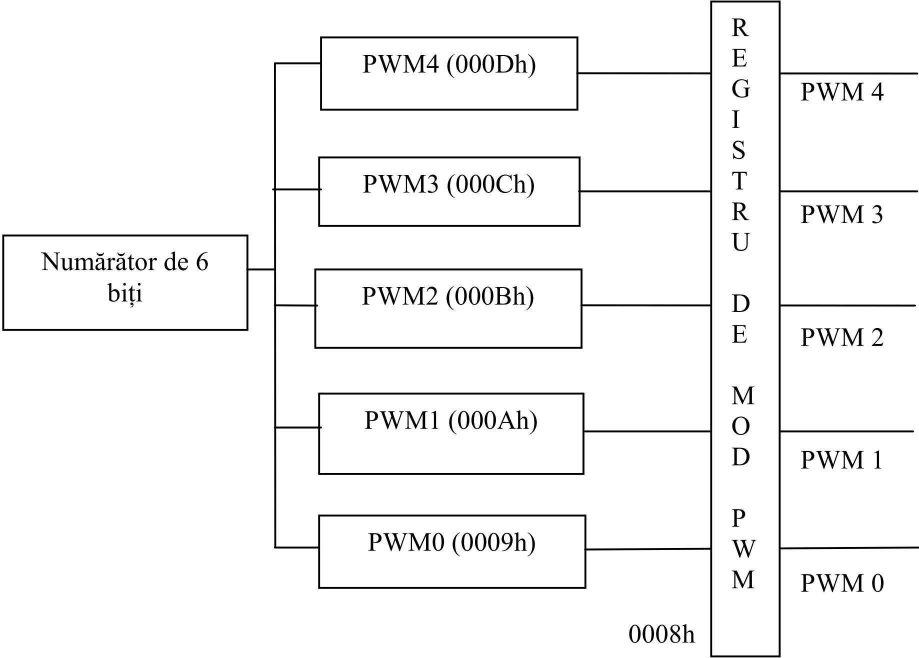


Figura 3.5 Structura modulului PWM la familia 6805

În registrul de mod sunt 5 biţi care validează modulul PWM astfel încât semnalele PWM să fie livrate pe la pinii portului D. Tot în registrul de mod mai este un bit, SCIB, care selectează dacă la adresa 000Dh să fie date pentru PWM sau rata de divizare pentru SCI. Dacă nu sunt folosite ca ieşiri PWM, liniile portului D pot fi folosite ca intrări/ieşiri obişnuite. Registrul de mod PWM se află la adresa 0008h.

Dacă se încarcă registrul de date cu 00h semnalul la ieşire va fi tot timpul zero. Înscrierea valorii 20h în registrul de date are ca rezultat un semnal cu factor de umplere 50% la ieşirea corespunzătoare, iar încărcarea valorii 3Fh va determina un semnal cu factor de umplere de 63/64.

f. Portul USB (MC68HC05JB4)

MC68HC05JB4 conţine un convertor A/D şi o interfaţă USB, ceea ce îl face ideal pentru achiziţia de date şi transferul lor către un calculator PC-AT. Datorită structurii USB, este posibilă realizarea unui punct central de prelucrare echipat cu PC şi multe puncte de achiziţie echipate cu acest MC.

Modulul USB implementează standardul USB 1.0, de mică viteză, la 1,5Mbps, cu 3 puncte de capăt (*End Point*). MC conţine şi transceiverul USB. Schema bloc a modului USB este dată în figura 3.6.

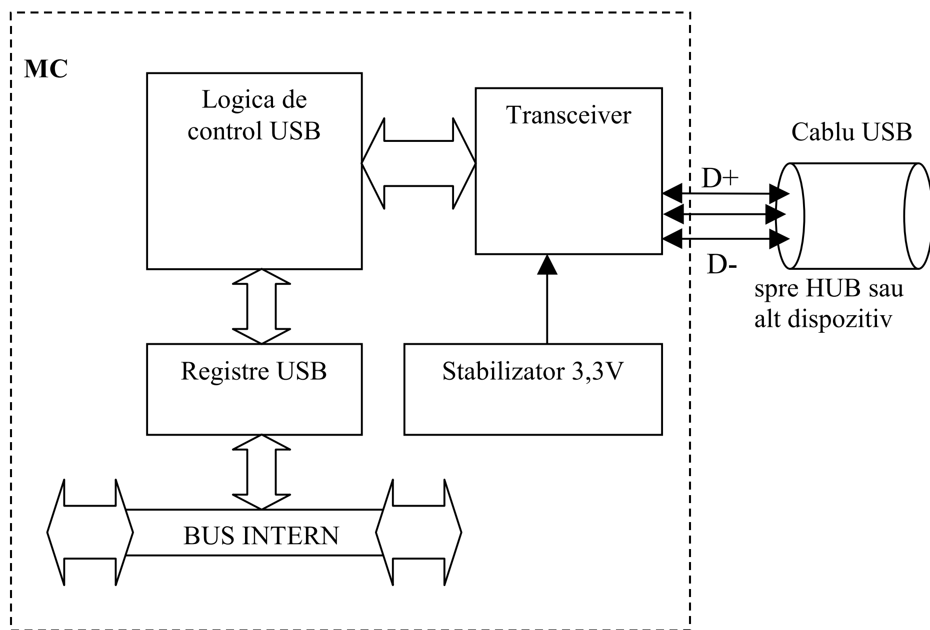


Figura 3.6 Structura internă a portului USB

Transceiverul are o ieşire diferenţială care poate lucra cu 3 stări pentru a permite transferul de date bidirecţional de tip full-duplex. Receptorul trebuie să fie de asemenea cu intrări diferenţiale.

Comanda interfeţei USB se realizează prin intermediul a 3 registre de comandă iar starea poate fi citită dintr-un registru de stare. Există un registru de adresă şi 24 de registre de date. Modul de funcţionare în întreruperi este programat prin registrul de întreruperi. Interfaţa USB poate lucra în modul de economie de energie dacă nu există trafic pe linii un anumit interval de timp. Apariţia unei date la recepţie sau emisia unei date de către UC produce ieşirea din starea inactivă.

g. Interfaţa LCD

MC6805 DragonKat este denumirea unui MC din familia 6805 care conţine o interfaţă pentru un afişor cu cristale lichide.

Interfaţa pentru LCD conţine:

- logica de control pentru sincronizare;
- RAM-ul de display care stochează datele pentru display printr-o corespondenţă 1 la 1 a biţilor (fiecare bit din memorie are corespondent un punct pe LCD). Memoria este organizată în cuvinte de 5 biţi şi poate fi scrisă sau citită;
- registru de date este folosit pentru a stoca datele din memoria RAM a LCD;
- generator de tensiune care conţine un divizor de tensiune pentru alimentarea segmentelor şi a planului din spate;
- driver pentru planul din spate;
- driver pentru segmente.

Structura unui afişaj LCD este arătată în figura 3.7.

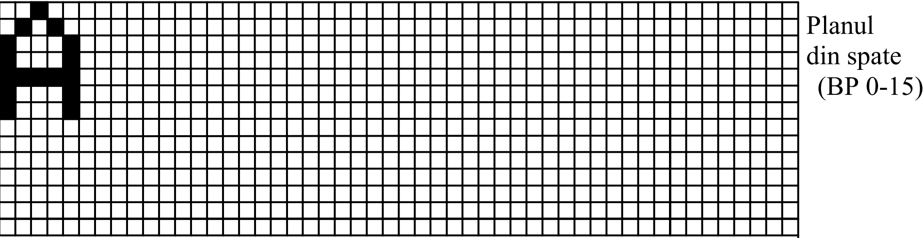


Figura 3.7 Afişaj LCD

Un punct este aprins când o linie Bpi şi una SEGi sunt active. Pentru caracterul A, memoria RAM de display are conţinutul din tabelul 3.4.

Tabelul 3.4

Conţinutul memoriei pentru afişarea caracterului A

Adresa	Bit0	Bit1	Bit2	Bit3	Bit4
200	0	0	1	0	0
201	0	1	0	1	0
202	1	0	0	0	1
203	1	0	0	0	1
204	1	1	1	1	1
205	1	0	0	0	1
206	1	0	0	0	1
207	0	0	0	0	0

Punctul de sus a lui A de exemplu, se scrie cu BP0 şi SEG2 active. Semnalele SEG şi BP sunt periodice şi baleiază tot timpul afişajul pentru a menține punctele aprinse. Frecvența semnalelor de aprindere a punctelor este de 32kHz, iar frecvența unui cadru este de 62,5Hz.

Dimensiunea panourilor LCD diferă, de aceea diferă şi consumul de curent. Tensiunea spre afişaj este obținută prin înserierea unor rezistențe, înseriere care poate fi comandată soft.

Schema bloc a interfeței pentru LCD este dată în figura 3.8

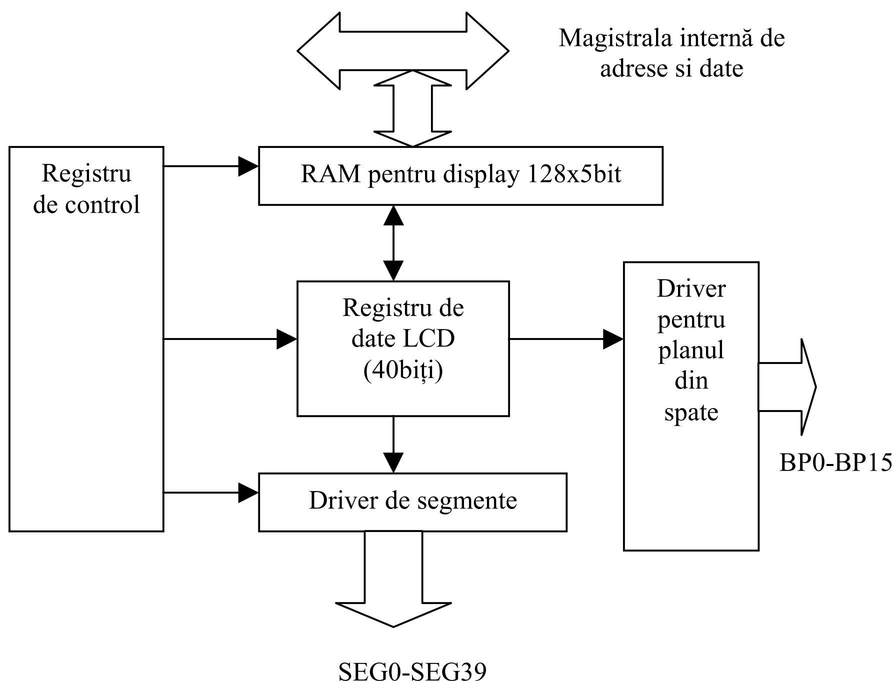


Figura 3.8 Schema bloc a intefeței LCD

3.1.7 Managementul puterii

Instrucțiunea STOP plasează UC în modul de lucru cu consum minim de energie. Ceasul intern este oprit şi toate procesele interne sunt oprite, inclusiv timer-ul. Întreruperile externe sunt validate automat în registrul CCR. Sunt şterse registrele aferente timer-ului şi canalului SCI, dar celelalte registre şi memoria nu sunt alterate. Unitatea centrală poate fi trezită din STOP doar de o întrerupere externă.

Instrucțiunea WAIT plasează UC într-un mod de lucru cu consum redus de energie. Rămân active timer-ul şi canalul serial SCI. Orice întrerupere validată, primită din exterior sau de la temporizator sau SCI, trezesc unitatea centrală.

Întreruperile sunt validate automat. Toate registrele şi memoria rămân cu datele avute la intrarea în starea WAIT.

3.1.8 Autoverificarea

MC are 2 moduri de operare: normal şi auto-verificare. Modul de operare este selectat de nivelul logic pe pinul IRQ în momentul unui RESET. În modul normal de operare începe rularea unui program din EPROM, deci EPROM-ul trebuie sa fie programat dacă a fost ales modul normal de operare. În modul autotest MC rulează un program conţinut în ROM în zona de memorie 3F00h-3FDEh. Autotestul comunică rezultatele la pinii PC3-PC0, unde se pot conecta LED-uri.

3.1.9 Programarea EPROM

Unele MC din familie conţin ROM, aşa cum este MC68HC05D9 care conţine 16K octeţi de PROM sau MC68HC05D24 care conţine 24K octeţi de ROM. Modelul MC68HC05D32 are 32K octeţi memorie EPROM, cu posibilitatea de ştergere UV. Înscrierea memoriei se face cu ajutorul unei tensiuni suplimentare de 15V (Vpp). Circuitele MC68HC705C8A conţin 8K EPROM sau OTP. Memoria EPROM sau OTP se poate programa cu ajutorul unui modul special, construit de Motorola, sau folosind un montaj în care progamul se încarcă în MC dintr-o memorie EPROM externă sau cu orice programator care poate să adreseze şi să pună date pe liniile de date (figura 3.9

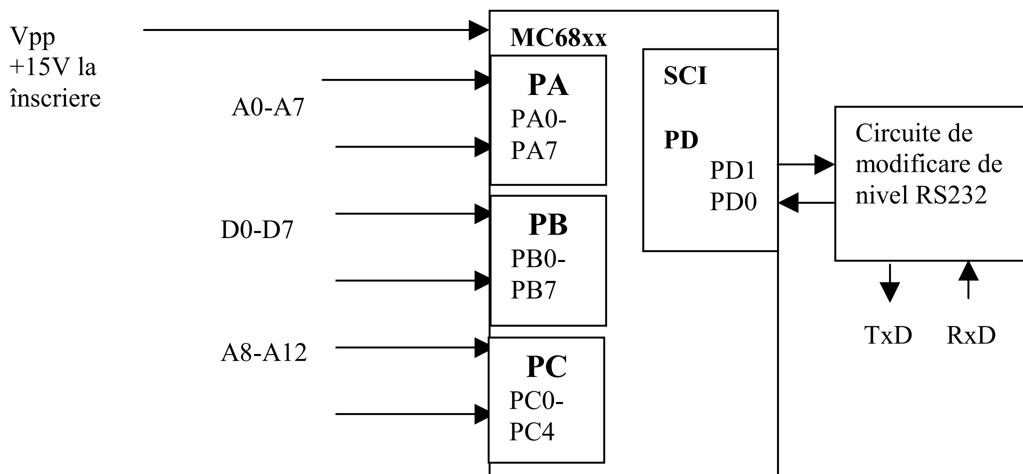


Figura 3.9 Schema unui programator EPROM (sau OTP)

Adresele se stabilesc la portul A, iar cele mai semnificative linii (A8-A12) se stabilesc la portul C (PC0-PC4). Liniile cu datele de înscris se stabilesc la portul B. Registrul folosit pentru programarea EPROM-ului este registrul de programare (PROG).

- Bit 2- LAT (*Latch Enable Bit*), poziționat în 1, comandă ca liniile de adrese şi date să fie bufferate pentru ca în următorul ciclu să se facă o scriere a EPROM-ului;
- Bit 0 (PGM) validează tensiunea Vpp pentru programarea EPROM-ului.

Se pune problema cum se poate ca un MC care nu a fost programat să lucreze sub comanda unui program ca să poată avea acces la programarea unui registru. Pentru programare MC trebuie să lucreze în modul bootstrap, mod în care MC intră prin forțarea liniei IRQ în timpul unui RESET. Programe simple utilizator pot fi introduse în RAM prin interfața SCI (*Serial Communications Interface*) şi rulate. Programul se transmite prin interfața serială, după care se rulează programul din RAM. Parametrii comunicației impliciți la Reset sunt 8 biți de date şi un bit de STOP, la viteza de 4800bps. Primul octet trimis reprezintă numărul total de octeți care vor fi trimişi. Conținutul EPROM-ului poate fi verificat prin citirea lui tot prin SCI. Conținutul EPROM-ului se poate ascunde, aşa încât să nu mai poată fi citit din exterior, prin poziționarea unui bit de securizare în registrul de opțiuni (adresa 1FDFh).

3.2 MC M68HC08

Familia 6808 urmează în timp familiei 6805, bineînțeles cu îmbunătățiri care vor fi amintite în acest subcapitol. MC din familia 6808 sunt compatibile cu cele din familia descrisă în subcapitolul anterior (3.1). Ca îmbunătățiri ale UC 6805 pot fi menționate:

- indicatorul de stivă este un registru de 16 biți (13 la 6805);
- registrul de index este de 16 biți, există posibilitatea manipulării separat a octetului superior şi a celui inferior (8 biți la 6805);
- frecvența internă CPU standard este de 8MHz (2MHz la 6805);
- se pot adresa 64K octeți memorie de date sau program (2K-16K la 6805),
- are 16 moduri de adresare (8 la 6805) şi 78 de noi coduri de instrucțiuni;
- se pot face mutări de date între locații de memorie fără intermediul acumulatorului;
- UC poate executa împărțiri de operanzi de 2 octeți la operanzi de 1 octet;
- operarea secvențială a UC poate fi oprită de două tipuri de evenimente: reset sau întreruperi. Sursele de Reset suplimentare față de 6805 sunt:

-detectarea unui cod de instrucțiune inexistentă;
-tensiune de alimentare sub o limită acceptată (LVI, *Low Voltage Inhibit*).

- sistemul de întreruperi admite un număr maxim de 128 de surse de întrerupere: reset, SWI şi IRQ0-IRQ125. Unele dintre aceste cereri de întrerupere sunt accesibile la pin. Întreruperea software are cea mai mare prioritate. În mod WAIT ceasul UC este oprit, dar celelalte module au ceas, astfel încât orice întrerupere trezeşte UC. În mod STOP toate ceasurile sunt oprite şi doar o întrerupere externă poate trezi UC.

Schema bloc a unui MC 6808 este prezentată în figura 3.10.

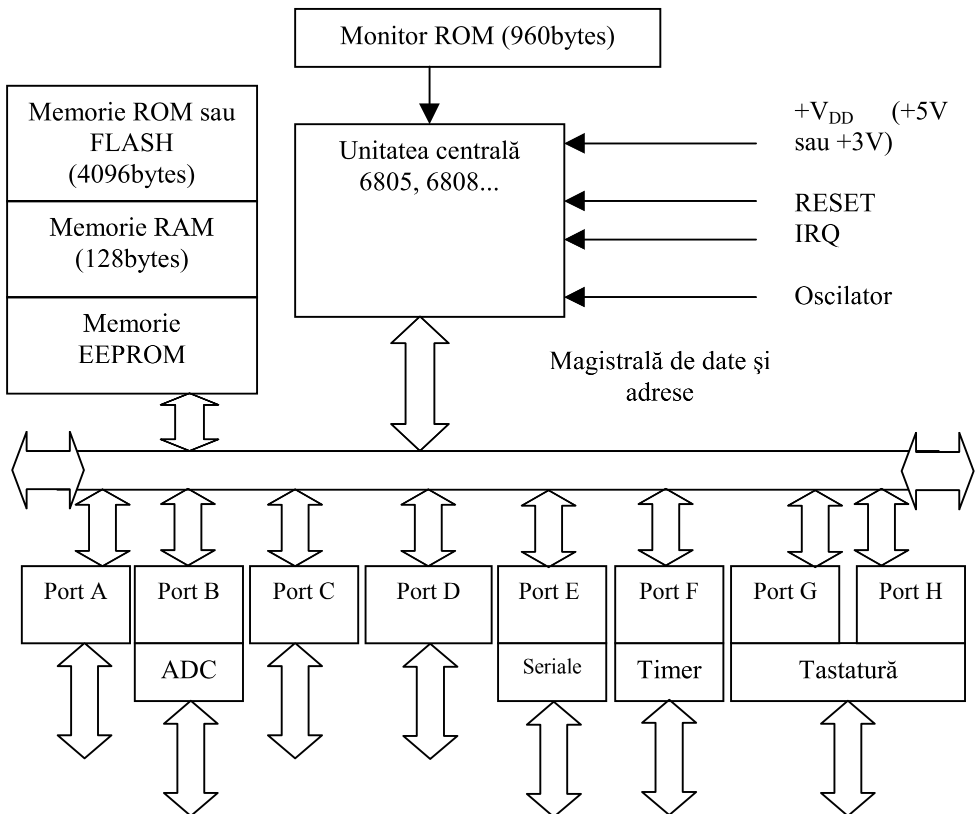


Figura 3.10 Schema bloc a unui MC din familia 6808

3.2.1 Unitatea centrală 6808

Unitatea centrală 6808 admite 2 moduri de lucru: un mod utilizator şi un mod monitor ROM. Modul de lucru poate fi alese prin poziționarea unui pin din exterior (IRQ) în 1 sau 0 în timpul unui RESET. Modul monitor creează posibilitatea legării MC cu un calculator gazdă printr-un canal serial, transferarea

software-ului de pe calculatorul gazdă în MC şi execuţia lui din RAM. Pentru MC echipate cu EEPROM în mod monitor se poate programa EEPROM-ul. Legătura serială între calculator şi MC este RS232 prin intermediul pinilor unui port I/O. În mod monitor MC poate să execute programe din RAM cu ajutorul unor comenzi simple monitor, toate funcţiile MC fiind valide. În modul monitor COP este invalidat. Transferul de date este bidirecţional. Fiecare comandă spre MC este urmată de un răspuns ecou al MC. Comenzile monitorului sunt:

- READ - citeşte o locaţie de memorie (se trimite cod+2 octeţi de adresă şi se returnează conţinutul locaţiei),
- WRITE - scrie o locaţie de memorie (se trimite cod+2 octeţi de adresă+octetul de scris)
- IREAD - citire indexată, citeşte următorii 2 octeţi din memorie faţă de ultima adresă accesată
- IWRITE - scriere indexată, scrie următorii 2 octeţi din memorie faţă de ultima adresă accesată
- READSP - citeşte indicatorul de stivă
- RUN - ruleaza program

Viteza de transfer implicită între MC şi calculator este de 4800bps cu pinul PTC3 ţinut la 1 în timpul resetului şi 9600bps cu pinul PTC3 ţinut la 0 în timpul resetului.

3.2.2 Interfeţe şi periferice on-chip

a. Porturi paralele I/O

MC 6808 poate gestiona 8 porturi I/O (A,B,C,D,E,F,G,H) cu linii care pot fi programate ca ieşiri sau intrări. Liniile nefolosite trebuie legate la masă sau la tensiunea de alimentare pentru a preveni defectarea circuitului prin descărcări electrostatice sau consumul excesiv de curent.

Fiecărui port îi corespunde un registru de direcţie (DDR) prin care se programează sensul de transfer al pinului. Unele porturi au semnificaţii duble, dacă liniile lor sunt folosite şi de alte interfeţe:

- portul B poate fi folosit ca 8 linii de intrare în convertorul A/D (ADC);
- portul C este un port de 6 biţi, bitul 2 se poate folosi pentru a furniza în exterior ceasul sistemului;
- portul D poate avea două linii folosite ca intrări pentru timer (Bit 6 şi Bit 4);
- portul E poate fi folosit ca şi canal SPI şi SCI;
- portul F este un port de 7 biţi care poate fi folosit ca intrări/ieşiri pentru timer;

- portul G este un port de 3 biți, liniile pot fi folosite ca interfață pentru tastatură;
- portul H este un port de 2 biți, liniile pot fi folosite ca interfață pentru tastatură.

b. Convertorul analog digital (ADC)

Convertorul A/D este un convertor pe 8 biți cu aproximări succesive care poate face conversii continuu sau la primirea unei comenzi. Sfârşitul conversiei poate fi semnalizat cu un bit indicator (flag) sau poate cere o întrerupere. Ceasul de conversie poate fi selectat. Un MC are 8 canale de conversie cu intrări multiplexate analogic.

Intrările pentru ADC se fac pe pinii portului de uz general PTB7-PTB0. Selecția canalului se face pentru a stabili care pin este intrare analogică. Ceilalți pini pot fi folosiți ca intrări/ieşiri de uz general. Dacă valoarea citită este VREFH, atunci valoarea digitală va fi FFh, iar dacă este VREFL atunci valoarea digitală va fi 00h. Aceste tensiuni, împreună cu tensiunea de alimentare pentru blocul analogic al MC sunt furnizate din exterior. Schema bloc a convertorului este dată în figura 3.11.

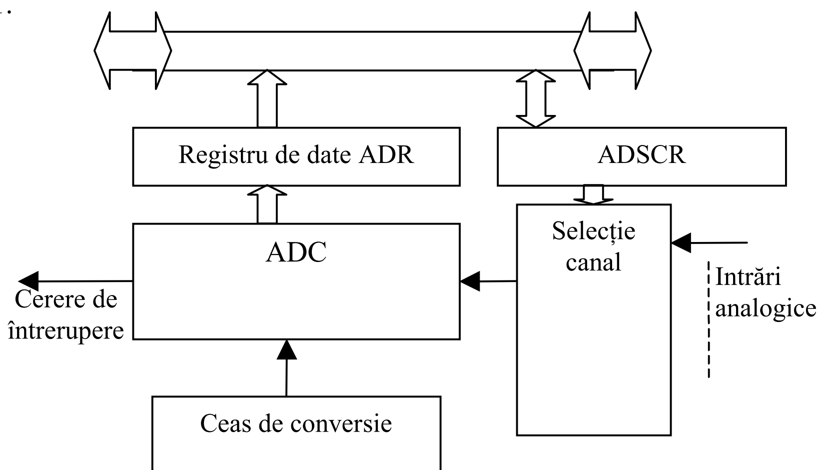


Figura 3.11 Schema bloc a convertorului analog digital

Registrele ADC sunt:

registru de control şi stare al ADC (ADSCR), care conține:

Bit 7 COCO/IDMAS, conversie completă sau selecție întrerupere DMA. Dacă întreruperile sunt invalidate (AIEN=0), COCO este poziționat de fiecare dată când se termină o conversie. În modul de conversie continuă datele se suprascriu în registrul de date după fiecare conversie şi COCO este poziționat doar după prima conversie. Dacă întreruperile sunt validate (AIEN=1), IDMAS selectează dacă întreruperea solicită şi un transfer DMA.

Bit 6 AIEN (ADC *Interrupt Enable*) validează întreruperile la sfârşitul unei conversii.

Bit 5 ADCO (ADC *Continous Conversion*), dacă este setat se face conversie continuă.

Bit4-0 ADCH4-0 selectează unul din cele 8 canale de conversie

registru de date (ADR)

registru de ceas (ADCLK) care conține:

Bit 7-5 ADIV2-0 formează rata de divizare a ceasului de conversie de la ADICLK (000) la ADICLK/16 (100).

Bit 4 ADICLK selectează ca ceas de conversie ceasul intern sau un ceas generat de timer.

O conversie începe după scrierea registrului ADSCR. Rezultatul conversiei este stocat în registrul de date (ADR) şi la terminarea conversiei se poziționează bitul COCO. Dacă se optează pentru conversie continuă, după prima conversie urmează alta, care rescrie registrul de date. Dacă sunt validate întreruperile, sfârşitul conversiei poate cere întrerupere către UC sau poate face şi o cerere DMA.

c. Interfața externă msCAN08 (Motorola Scalable CAN)

Protocolul CAN a fost definit de BOSCH în 1991 pentru utilizarea pe o magistrală la autoturisme, unde să îndeplinească condiții specifice: procesare în timp real, fiabilitate într-un mediu perturbat şi preț mic. Caracteristici ale magistralei msCAN08 sunt:

- transmisie serială sincronă cu blocuri între 0-8 octeți;
- viteza de transfer până la 1MBps;
- transferul serial se face pe 2 linii, o intrare (RxCAN) şi o ieşire TxCAN).

Cuplarea la magistrală nu se face direct, ci prin transceiver-e care pot suporta un curent important şi care pot detecta dacă un MC are o linie defectă care ar forța magistrala la conectare directă (figura 3. 12).

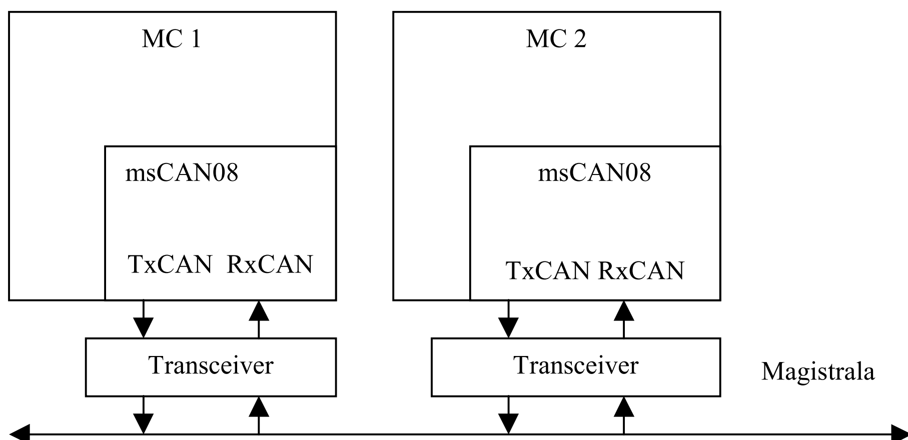


Figura 3.12. Cuplarea la magistrala CAN

O implementare modernă trebuie să respecte două condiții:

- orice dispozitiv CAN poate să transmită un şir de mesaje fără să elibereze magistrala între mesaje. Arbitrarea magistralei se va face doar după ce s-a terminat transmiterea mesajului.
- mesajele sunt astfel organizate încât cel mai prioritar mesaj va fi trimis înaintea celor care stau de mai mult timp în coada de aşteptare.

Aceste cerințe pot fi îndeplinite cu o schemă de transmisie cu buffere duble. Chiar şi aşa, unitatea centrală nu are întotdeauna timp să aranjeze datele în buffer, aşa încât la 6808 s-a realizat o arhitectură cu 3 buffere.

Mesajul recepționat este stocat într-o stivă FIFO cu 2 nivele. Există 2 buffere de recepție de 13 octeți aranjate într-o singură arie de memorie, astfel încât registrul de date recepționate are o singură adresă. Registrul RxBG *(Background Receive Buffer*) primeşte datele seriale şi le transferă în *RxFG (Foreground Receive Buffer*) care poate fi adresat de unitatea centrală. Dacă mesajul recepționat este corect (are un identificator valid) se poziționează bitul RxF (*Receiver Full Flag*) şi se generează o cerere de întrerupere. În acest timp în registrul RxBG se recepționează un nou cadru. Schema bloc pentru recepție şi transmisie este dată în figura 3.13.

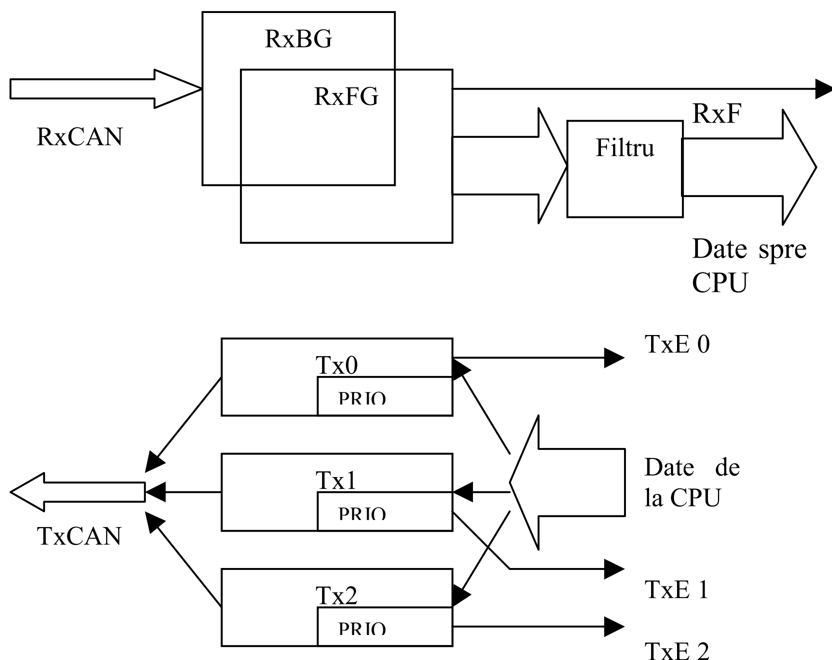


Figura 3.13 Schema bloc de receptie, emisie CAN

Pentru a micşora numărul de întreruperi solicitate UC de către receptor s-a introdus un filtru de acceptare. Datele msCAN au o anumită structură, primii biți

fiind de identificare a cadrului şi de adresă a destinației. Este posibil ca interfața CAN să verifice aceşti primi biți şi să facă transferul din RxBG în RxFG numai dacă mesajul este destinat acestui MC.

Când ambele registre de recepție sunt pline şi se recepționează date se generează o eroare de suprascriere. Se abandonează recepția noilor caractere, se semnalizează eroarea, dar transmițătorul rămâne funcțional.

Transmiterea mesajelor se face cu o structură de 3 buffere, fiecare de 13 octeți. Un registru suplimentar TBPR (*Transmit Buffer Priority Register*) stabileşte prioritatea mesajelor. Fiecare buffer de transmisie semnalează UC faptul că este gol prin poziționarea unui bit TXE (*Transmit Buffer Empty*) în registrul de stare CTFLG (*CAN Transmitter Flag Register*). Bitul TXE poate cere o întrerupere pentru ca UC să poată reîncărca bufferul de transmisie care s-a golit. Dacă mai mult de un registru este plin şi gata de transmisie, se face apel la registrul de priorități pentru arbitrare. Fiecare buffer de transmisie are o zonă de 8 biți (PRIO) în care programul utilizator scrie prioritatea mesajului când se transmite mesajul de la UC (cea mai mică valoare reprezentând cea mai mare prioritate).

Dacă se transmite un mesaj cu prioritate mai mare şi se doreşte abandonarea mesajului curent, se solicită aceasta cu bitul ABTRQ *(Abort Request Flag*) în registrul de control al transmisiei CTCR (*Transmission Control Register*). Dacă solicitarea de abandonare este posibilă, interfața CAN va poziționa bitul ABTAK (*Abort Request Acknowledge*) şi TXE pentru a se putea transmite mesajul prioritar.

d. Interfața cu tastatura

Interfața cu tastatura constă în 5 linii independente de intrare în MC care pot solicita întreruperi mascabile. Funcționarea acestei interfețe este asistată de următoarele registre:

registrul de stare şi control al tastaturii (KBSCR) conține:

Bit 3 - indicatorul existenței unei întreruperi, KEYF. Bitul este 1 dacă se află în curs de servire o întrerupere de la tastatură;

Bit 2 - confirmare, ACKK. Dacă acest bit este poziționat se şterge bufferul întreruperilor de la tastatură;

Bit 1 - validarea întreruperilor de la tastatură, IMASKK. Dacă acest bit are valoarea 0 sunt validate întreruperile;

Bit 0 - modul de întrerupere, MODEK. Dacă bitul este 1 se poate cere o întrerupere cu un front căzător sau cu un nivel 0; dacă bitul este 0 se poate cere o întrerupere doar cu front căzător.

registrul de validare a întreruperilor de la tastatură care cu biții

Bit 4 - Bit 0 validează pinul portului I/O ca cerere de întrerupere de la tastatură sau ca port standard.

3.2.3 Programarea memoriei EEPROM

Circuitele din familia 6808 pot avea 512 octeți de memorie EEPROM care poate fi ştearsă şi reînscrisă fără o tensiune externă suplimentară. Memoria programată poate fi protejată împotriva unor scrieri sau ştergeri accidentale.

Starea unui bit neprogramat este 1 logic. Programarea lui înseamnă trecerea în stare 0. Programarea pote fi făcută în mod redundant, adică primii 256 de octeți să fie identici cu următorii.

Registrele EEPROM sunt:

registrul de control al EEPROM EEPROM Control Register **(EECR):**

Bit 7 – EEBCLK, validare ceas intern pentru EEPROM; selectează cesul folosit pentru programarea EEPROM: ceasul intern sau un oscilator RC intern.

Bit 5 – EEOFF; invalidează EEPROM-ul în modurile de lucru cu economie de energie.

Bit4, Bit 3 - EERAS 1, EERAS 0; stabilesc operația care se execută conform tabelului 3.5.

Tabelul 3.5

Programarea operațiilor în registru de control

EEBPx	EERAS1	EERAS0	Operația
0	0	0	Programare de octet
0	0	1	Ştergere de octet
0	1	0	Ştergere bloc
0	1	1	Ştergere globală
1	X	X	Nu se executa programare/stergere

Bit 2 - EELAT; dacă este setat, comandă memorarea datelor şi adreselelor pentru scriere iar dacă este 0, se face citire din EEPROM.

Bit 0 – EEPGM; dacă este setat, validează scrierea sau ştergerea. Poziționarea lui EEPGM trebuie să fie precedată în alt pas de poziționarea lui EELAT.

registrul EEPROM nevolatil (EENVR):

Bit 7 – EERA, arie redundantă; configurează memoria în două jumătăți redundante.

Bit 4 - CON0; stabileşte dacă se validează securitatea memoriei (cu 0) sau nu.

Bit 3-0 – EEBP3-0, biți de protecție; dacă aceşti biți sunt 1, blocul de memorie este protejat la scriere.

registrul de configurație a ariei, EEACR se poate citi ceea ce a fost înscris în registrul EENVR. Din rațiuni de siguranță a informației, datele de configurare a

EEPROM se scriu într-un registru nevolatil, EENVR. La un RESET, datele din EENVR se scriu într-un registru volatil corespondent (EEACR).

În modul de economie de energie STOP nu se poate face înscrierea sau ştergerea memoriei EEPROM.

Pentru înscrierea EEPROM se parcurg următorii paşi:

- se resetează EERAS1 şi EERAS0 (şi se setează EELAT în EECR)
- se scrie octetul în EEPROM
- se setează EEPGM
- se aşteaptă un timp pentru ca programarea să poată fi executată
- se resetează EEPGM
- se aşteaptă un timp pentru ca tensiunea de programare să scadă
- se resetează EELAT

O ştergere pe bloc sau globală se face cu biţii EERAS ca în tabelul 3.5, orice adresă din bloc fiind posibilă. Împărţirea în blocuri este EEBP0 (0800h-087Fh), EEBP1 (0880h-08FFh), EEBP2 (0900h-097Fh), EEBP3 (0980h-09FFh). În modul redundant EEBP2 şi 3 nu au nici o semnificaţie.

3.3 MC PE 16 BIŢI - 6816

Unitatea centrală 6816 este o unitate centrală cu magistrala de date de 16 biţi şi o magistrală de adrese pe 20 de biţi care permite adresarea a 1Moctet de memorie. UC conţine 2 registre acumulator de 16 biţi şi 3 registre index de 16 biţi pe lângă registrele clasice. Unitatea centrală poate executa instrucţiuni pe 8 biţi, pe 16 biţi sau pe 32 de biţi. Toate instrucţiunile de la UC 6811 pot fi executate de 6816, dar cu alt număr de tacte, deci cu altă viteză. Unitatea centrală funcţionează la maximum 16MHz. Tehnologia de fabricaţie permite funcţionarea şi la viteze mai mici, chiar în regim static (ca şi celelalte UC). Unitatea centrală conţine facilităţi DSP (*Digital Signal Processing*), fiind posibile înmulţiri cu numere fracţionare. UC permite 10 tipuri de adresare, din care 6 sunt preluate de la 6811.

Pentru păstrarea compatibilităţii, registrul PC şi indicatorul de stivă sunt de 16 biţi, dar apare un registru nou, numit K, ce reţine extensiile de la 16 biţi la 20 de biţi pentru registrul PC şi SP. În structura UC mai apar şi registrele pentru înmulţiri, 2 registre de 16 biţi (H şi I) şi unul de 36 de biţi pentru rezultat (MAC).

Pentru comunicaţia între MC sau pentru adresarea memoriei externe s-a standardizat un bus extern, numit IMB (*Intermodule Bus*), cu 16 linii de date şi 24 linii de adresă, la care s-a aliniat şi 6816. Acest bus este de tip asincron, permiţând transfer pe 8 sau 16 biţi în urma unui protocol (*handshake*).

3.3.1 Modulul de integrare (SIM, *System Integration Module*)

Acest modul, apărut de la 6811, grupează blocurile care controlează sistemul şi cuprinde:

- Modul pentru configurarea sistemului şi protecţie:
    1. un monitor de bus; resetează sistemul dacă apar cicluri de magistrală mai lungi de 8-64 tacte de ceas sistem;
    2. un monitor de HALT; poziţionează un bit în registrul de stare al RESET-ului când a apărut o instrucţiune de HALT pe bus;
    3. un monitor de întreruperi; supraveghează apariţia unei arbitrări în timpul unui ciclu de întrerupere;
    4. un ceas de gardă (watch dog);
- Modul pentru sintetizarea ceasului; poate fi realizată din mai multe surse; dintr-o sursă internă cu ajutorul unei bucle PLL, dintr-o sursă externă cu ajutorul buclei PLL sau direct dintr-o sursă externă.
- Modul de formare a bus-ului extern
- Modul de test

3.3.2 Interfeţe

Interfeţele seriale sunt grupate sub forma unui modul serial cu coadă de aşteptare (QSM *Queued Serial Module*) care conţine o interfaţă serială SPI şi una SCI. Coada de aşteptare este formată în RAM şi are dimensiunea de 16 cuvinte de 8-16 biţi fiecare. Transmisia acestui şir de date poate fi executată automat, fără intervenţia unităţii centrale. Modulul QSM este cuplat pe bus-ul IMB.

Convertorul analog numeric (ADC) este un convertor cu aproximaţii succesive cu rezoluţia programabilă de 8-10 biţi, cu 8 canale multiplexate. Convertorul este conectat pe bus-ul IMB.

Pentru date importante (cum ar fi cele din stivă) există o memorie RAM static de 1K octet care la căderea tensiunii de alimentare este alimentată de la un pin special, unde se poate conecta o baterie.

Timer-ul are 11 canale, din care 2 sunt folosite pentru generarea semnalelor PWM.

Circuitul este capsulat în capsule de 132 sau 144 pini.

Schema bloc a circuitului este dată în figura 3.14.

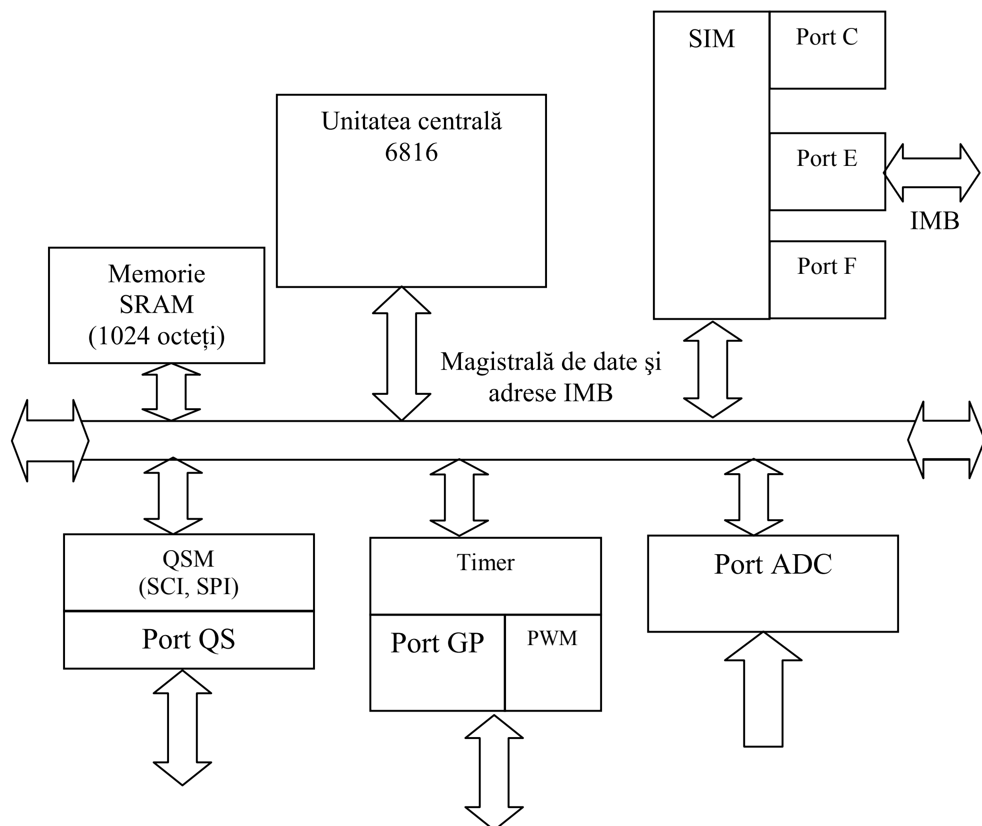


Figura 3.14 Schema bloc MC 6816

3.4 MC PE 32 DE BIȚI - 68300

MC 68300 este asemănător arhitectural cu MC pe 16 biți. În jurul UC sunt grupate pe magistrala IMB un modul de integrare, un convertor analog digital cu 8 canale, un modul de comunicație serială cu coadă de aşteptare (QSM), 512 octeți RAM static, 3,5K octeți RAM care pot fi folosiți şi de timer (TPURAM), un modul de timp (TPU, *Time Processor Unit*), până la 6 porturi I/O de uz general şi 2 module flash EEPROM.

Ceasul intern al sistemului este de maximum 16MHz.

Unitatea centrală CPU32 este compatibilă soft cu procesoarele din familia 68000. Ea conține 16 registre generale de 32 de biți cu care se poate lucra pe 8, 16 sau 32 de biți, un PC de 32 de biți, un SP de 32 de biți, un registru de stare şi alte registre.

Timer-ul este o unitate microprogramată care poate lucra separat de UC, şi care conține propriul RAM. Timer-ul este format din 16 canale independente. Fiecare poate executa o funcție de timp şi poate fi programat independent.

Memoria flash EEPROM serveşte la stocarea nevolatilă a informațiilor cum ar fi rutine ale sistemului de operare sau date care sunt apelate frecvent. Memoria EEPROM este foarte rapidă, ea poate răspunde în 4 tacte. MC este echipat cu 2 module de flash EEPROM, unul de 16K octeți şi unul de 48K octeți. Memoria EEPROM poate fi configurată pentru a lucra în modul bootstrap. Memoria EEPROM poate fi ştearsă doar neselectiv (în întregime) şi are nevoie de o alimentare specială.

3.5 DATE COMPARATIVE PENTRU MC MOTOROLA –CISC

Tabelul 3.6 arată principalele performanțe, comparativ, pentru MC din familia Motorola:

Tabelul 3.6

Tabel comparativ pentru MC Motorola - CISC

MC	Magistrala de date	Frecvență (MHz)	Linii I/O	Interfețe speciale	Preț (USD)
6805	8	2	31	6	1.9
6808	8	8	48	6	2.1
6816	16	16	48	6	8.25
68300	32	16	48	6	17.5

4

FAMILIA MCS-51

8051 a fost propus de INTEL în a doua generaţie de MC şi este cel mai folosit şi cel mai bine vândut MC din lume. În 1976 Intel a prezentat familia de microcontroller-e MCS48 care este compusă din 8048, 8748 şi 8035. A apărut astfel pentru prima dată pe piaţă un microcalculator complet pe un singur chip. MC includea o unitate centrală pe 8 biţi, memorie ROM sau EPROM de 1024x8biţi, RAM 64x8 biţi, porturi I/O şi timere. În etapa a doua, INTEL a lansat familia MCS-51 formată din MC 8051, 8751 şi 8031. INTEL recomandă ca punerea la punct a aplicaţiei să fie realizată pe MC 8751 care este prevăzut cu memorie EPROM (4K octeţi). Memoria poate fi programată cu orice programator, dar se recomandă utilizarea programatorului de la INTEL (UPP, Universal PROM Programmer). Pentru produse în serie mare se recomandă 8051, care are memorie ROM, programată de furnizor la comanda beneficiarului şi care revine la un preţ de cost mult mai mic. MC 8031 nu conţine ROM, dar se poate ataşa în exterior ROM, PROM sau EPROM, direct sau multiplexat.

8051 este un MC foarte puternic şi uşor de programat. Ca dovadă, în tabelul 4.1 se prezintă câteva modele oferite de diferiţi furnizori, echivalente cu 8051:

Tabel 4.1

Modele echivalente 8051 oferite de diferiţi producători

Varianta	Nr. pini	Produ-cător	RAM	Mem. program	Observaţii
8031	40	Toţi	128	64K	Procesor de bază, UART, Tmr0+1
8051	40	Toţi	128	4K	Procesor de bază, UART, Tmr0+1
80C51GB	68	Intel	256	64Kx	8051+PCA, 8bA/D
87C51GB	68	Intel	256	8K	8051+PCA, 8bA/D
SABC502	40	Siemens	256	64Kx	8052+XRAM+WDog
80C552	68	Philips	256	64Kx	10bA/D, i2c, CC,PWM
83CL580	56,	Philips	256	6K	LV 8052+A/DC+i2c+WDog
80C320	40	Dallas	256	64Kx	Fast, 2UART
89S8252	40.	Atmel	256	10K	FLASH 8k, WDog
89C55	40.	Atmel	256	20KF	FLASH,fast, LV

PCA- arie programabilă de numărătoare (programmable counter array)
LV- tensiune redusă 3,3V (low voltage)
PWM- modularea impulsurilor în lăţime (pulse width modulation)
CC- intrare şi comparare (capture/compare)
i2c- Interfaţa I^2C (Philips)
Tmr- timer

Un tabel cu mai multe exemple este disponibil pe discul care însoţeşte cartea.

4.1 STRUCTURA ŞI FUNCŢIONAREA

Schema bloc internă este dată în Fig. 4.1.

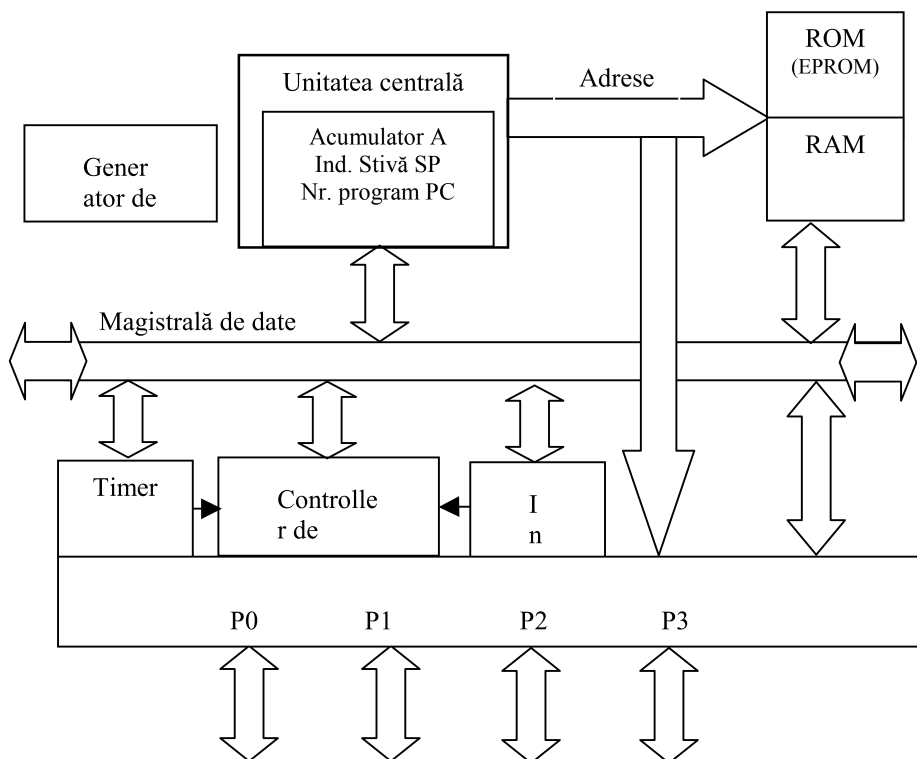


Figura 4.1 Schema bloc a unui MC 8051

Unitatea centrală este o unitate aritmetică şi logică pe 8 biţi care conţine următoarele registre:

- acumulator A - 8 biți;
- registrul de stare PSW (Program Status Word) - 8 biți;
- registrul B (8 biți) folosit în operațiile de înmulțire şi împărțire;
- indicatorul de stivă (Stack Pointer) - 8 biți;
- un registru numărător de date (DPTR Data Pointer) - 16 biți - care poate fi manipulat şi ca 2 registre de câte 8 biți (DPH şi DPL) şi serveşte ca bază în salturi indirecte sau în transferuri externe;
- numărătorul de program (PC Program Counter) - 16 biți.

Memoria pentru program (ROM) este separată şi distinctă de memoria pentru date, are alte mecanisme de adresare şi alte semnale de comandă. MC poate lucra atât cu memoria pe chip cât şi cu memorie externă; există astfel posibilitatea de a mări capacitatea memoriei de lucru. Memoria externă este adresată automat dacă adresa din instrucțiune este în afara zonei de memorie internă.

Familia 8051 are 32 de linii I/O, configurate ca 4 porturi de 8 biți. Fiecare linie poate fi programată individual ca linie de intrare, de ieşire sau bidirecțională. Unele linii au semnificații alternative. Portul 0 este folosit şi ca magistrală multiplexată de adrese şi date, iar pentru transferuri care au nevoie de adrese de 16 biți, se foloseşte portul P2 pentru octetul cel mai semnificativ de adresă. Liniile portului 3 sunt folosite ca cereri de întrerupere (2 linii), intrări/ieşiri pentru timer (4 linii) şi linii seriale (2).

8051 are 2 timere de 16 biți, fiecare poate fi programat separat. Timerele pot fi utilizate pentru a măsura intervale de timp, pentru a determina lungimea unor impulsuri, ca numărătoare, etc. Rezoluția este de 1μs, intervalul maxim de timp fiind de 65,536ms.

Portul serial UART este un port serial full duplex cu rata de transfer până la 115Kbaud. Portul serial poate fi folosit atât pentru comunicații cu echipamente periferice cât şi comunicații între circuite.

Controllerul de întreruperi admite întreruperi de la 5 surse:

- de la portul serial (dacă s-a transmis sau recepționat un caracter);
- de la timere (când s-a înregistrat o depăşire);
- de la 2 pini de intrare.

Fiecare sursă poate fi validată sau invalidată individual. Prioritatea poate fi programată. Fiecare sursă are asociată o adresă în memoria program (tabelul 4.2).

Tabelul 4.2

Adresele de salt pentru întreruperi

Sursa de întrerupere	Adresa de salt	Sursa de întrerupere	Adresa de salt
RESET	0000h	Extern 1	0013h
Extern 0	0003h	Timer 1	001Bh
Timer 0	000Bh	Port serial	0023h

4.1.1 Descrierea semnalelor la pini

Vcc- tensiunea de alimentare, +5V;
Vss- masă;
PORT 0[*] - este un port pe 8 biți bidirecțional cu drena în gol. Este port de date şi adrese (octetul cel mai puțin semnificativ) pentru memoria externă. De asemenea portul 0 primeşte octeții pentru programarea EPROM-ului intern iar în timpul verificării programului din EPROM, datele pot fi citite tot prin portul 0.
PORT 1*- este un port pe 8 biți bidirecțional, cu drena în gol. În timpul programării şi testării EPROM-ului, la portul 1 se stabileşte octetul cel mai puțin semnificativ de adresă. Pinii P1.0/T2 şi P1.1/T2X au funcții duble. Astfel T2 este intrare externă în Timerul 2 iar T2X este intrarea de comandă a unei capturi în Timer-ul 2.
PORT 2*- este un port de 8 biți bidirecțional. În timpul programării şi testării EPROM-ului la portul 2 se stabileşte octetul cel mai semnificativ de adresă. Pentru adresarea memoriei externe pe 16 biți portul 2 generează partea mai semnificativă a octetului de adresă. La adresare memoriei externe pe 8 biți portul 2 este registrul cu funcții speciale P2.
PORT 3*- este un port de 8 biți bidirecțional. Pinii portului 3 au semnificație alternativă (tabelul 4.3).

Tabelul 4.3

Semnificația dublă a pinilor de la PORT 3

Pin	Semnificație
P3.0	RXD-intrare serială a porului pentru recepție
P3.1	TXD-ieşire serială a portului pentru emisie
P3.2	/INT0-întrerupere externă pentru Timer 0
P3.3	/INT1-întrerupere externă pentru Timer 1
P3.4	T0-intrare externă în Timer 0
P3.5	T1-intrare externă în Timer 1
P3.6	/WR-strob pentru scrierea memoriei de date externă
P3.7	/RD-strob pentru citirea memoriei de date externă

RST este o intrare de RESET. Dacă semnalul de intrare rămâne în HIGH pe perioada a doi cicli maşină în timp ce oscilatorul funcționează, are loc inițializarea MC.
ALE/PROG ALE (Address Latch Enable) validează octetul cel mai puțin semnificativ de adresă în timpul accesului la memoria externă (PORT0). PROG

[*] Ieşirile sunt bufferate şi pot suporta până la 4 intrări LS TTL. Pinii programați ca ieşiri în stare HIGH pot fi folosiți ca intrări. Ca pini de intrare, dacă au fost forțați în LOW din exterior, aceştia furnizează curent.

este semnal de intrare pentru impulsul de programare în timpul programării EPROM-ului.
/PSEN (Program Store Enable) validează citirea programelor din memoria program externă. Nu se activează la citirea datelor din memoria externă de date.
/EA/VP /EA (External Acces Enable) validează în stare HIGH memoria program internă, iar în stare LOW validează memoria program externă. La VP se aplică tensiunea de 21V în timpul programării EPROM-ului.
XTAL1, XTAL2 se conectează cristalul de cuarț sau un oscilator extern.

4.1.2 Gestionarea memoriei

MC8051 au spații diferite de adresare pentru memoria program şi memoria de date. Spațiul maxim de adresare este de 64K atât la memoria program cât şi la cea de date. Un extras din datele de catalog arată tipurile şi dimensiunea memoriei pentru diferite circuite din familie (tabelul 4.4).

Tabelul 4.4

Echiparea cu memorie a diferitelor circuite 8051

MC	Memorie internă de program	Memorie internă de date
8031AH	-	128bytes RAM
8051AH	4Kx8 ROM	128bytes RAM
8751H	4Kx8 EPROM	128bytes RAM
8032AH	-	256bytes RAM
8052AH	8Kx8 ROM	256bytes RAM
8752BH	8Kx8 EPROM	256bytes RAM

a. Gestionarea memoriei de date (RAM)

Memoria internă este mapată în spațiul de adresare 00H - FFH (256bytes). Ea poate fi "umbrită multiplu"; aceeaşi adresă poate fi atribuită mai multor arii de memorie (internă SRAM, internă extinsă SRAM, ariei registrelor programabile, memoriei externe de program sau de date). Umbrirea este realizată fie utilizând semnale hard (PSEN, nRD, nWR), fie utilizând instrucțiuni specifice ce se referă strict la unul dintre tipurile de memorie (MOVC sau MOVX). În zona 80H -FFH sunt registrele speciale (SFR) care sunt adresabile direct. Zona 00H-7FH poate fi adresată direct sau indirect. Memoria externă se află între adresele 0000H-FFFFH. Validarea folosirii memoriei de date se realizează cu semnalele nRD şi nWR. Aceste semnale (nRD şi nWR) se folosesc şi pentru selecția datelor din portul 0, unde datele sunt multiplexate cu adresele. Adresele sunt validate cu semnalul ALE. Informația poate fi accesată cu adresare pe 8 sau pe 16 biți.

b. Gestionarea memoriei program (ROM, EPROM)

Se poate folosi memoria ROM internă (nEA în stare HIGH) sau cea externă (nEA în stare LOW). Strobul pentru citirea memoriei externe este semnalul PSEN. Toate citirile se fac cu adresare pe 16 biţi.

4.1.3 Circuitele timer

MC 8051 are 2 numărătoare de 16 biţi, iar 8052 3 numărătoare. În modul de funcţionare ca temporizator registrul este incrementat la fiecare ciclu cu un impuls la 1/12 din frecvenţa oscilatorului. În modul de funcţionare ca numărător, registrul este incrementat la fiecare tranziţie din 1 în 0 la pinul extern corespunzător T0, T1 şi T2. Rata maximă de numărare este 1/24 din frecvenţa oscilatorului.

Programarea funcţionării timer-elor se face cu registrul TMOD din SFR, cu structura: pentru numărătorul 1: GATE1, C/-T1 - M11, M01,
pentru numărătorul 2: GATE2, C/-T2 - M12, M02, unde:

GATE - gestionează controlul. Când GATE=1, timer-ul este activat doar cât timp pinul nINT corespunzător este în HIGH şi şi bitul TR din registrul SFR (TCON) este 1.

C/-T - selectează operarea ca timer (LOW) sau counter (HIGH)

M1 şi M0 au semnificaţia din tabelul 4.5.

Tabelul 4.5

Programarea modului de operare cu biţii M0 şi M1

M1	M0	Modul de operare
0	0	Timer pe 13 biţi (Mod 0)
0	1	Timer/Counter pe 16 biţi (Mod 1)
1	0	TL0 este timer/counter pe 8 biţi şi TH0 este timer pe 8 biţi (Mod 2)
1	1	Timer/Counter 1 oprit (Mod 3)

Semnificaţia biţilor din registrul special TCON este:

TF1, TR1, TF0, TR0, IE1, IT1, IE0, IT0, unde:

TF0, TF1- indicatorul de depăşire al timer-ului 0, respectiv 1. Este setat la depăşire ca numărător sau teporizator şi este resetat când se apelează rutina de întrerupere a timerului corespunzător;

TR0, TR1-este setat sau resetat pentru a porni sau pentru a opri numărătoarele/ temporizatoarele;

IE0, IE1 - indicator de activare a întreruperii pe front;

IT0, IT1 - controlul activării întreruperii (pe nivel IT=0, sau pe front IT=1).

Pentru MC care au al treilea timer există registrul T2CON cu structura:

TF2, EXF2, RCLK, TCLK, EXEN2, TR2, C/-T2, CP/RL2, unde:
TF2 - indicator de depăşire;
EXF2 - indică apariţia unui eveniment la pinii de control extern;
RCLK - când este setat, portul serial foloseşte ca şi tact de recepţie semnalul de depăşire generat de timerul 2;
TCLK - când este setat, portul serial foloseşte ca şi tact de emisie semnalul de depăşire generat de timerul 2;
EXEN2- indicator de activare externă;
TR2 - porneşte/ opreşte timerul 2
C/-T2 - selectează funcţionarea ca timer sau numărător;
CP/RL2 - indicator pentru memorare.

4.1.4 Interfaţa serială UART

Portul serial este de tip asincron, full-duplex. Portul serial poate opera în 4 moduri:

Modul 0: datele seriale sunt transferate în ambele sensuri prin RXD. TXD furnizează ceasul de transmisie. Rata transferului este 1/12 din frecvenţa de oscilaţie.
Modul 1: transmisie asincronă, se transmit date prin TXD, se recepţionează prin RXD în formatul 1 bit de start, 8 biţi de date, 1 bit de stop, cu rata de transfer variabilă (programabilă) cu un timer.
Modul 2: ca la modul 1, dar se transmite şi un al 9-lea bit de date care poate fi bitul de paritate, cu rata de transfer egală cu frecvenţa oscilatorului divizată cu 32 sau cu 64.
Modul 3: ca la modul 2, cu rata de transfer variabilă (programabilă) cu un timer.

Registrul de control al portului serial SCON (din registrele SFR):
SM0, SM1, SM2, REN, TB8, RB8, TI, RI;

SM0, SM1 programează modul de lucru al portului serial conform tabelului 4.6.

Tabelul 4.6

Programarea modului serial cu bitii SM0 si SM1

SM0	SM1	Mod
0	0	Mod 0
0	1	Mod 1
1	0	Mod 2
1	1	Mod 3

SM2 - activează posibilitatea de lucru multiprocesor în modurile 2 şi 3, prin transmisia bitului special 9;
REN - activare/ dezactivare recepţia;
TB8 - al 9-lea bit ce se transmite în modurile 2 şi 3;
RB8 - al 9-lea bit ce se recepţionează în modurile 2 şi 3;
TI - indicator de întrerupere a transmisiei;
RI - indicator de întrerupere a recepţiei.

4.1.5 Sistemul de întreruperi

MC 8051 dispun de 5 nivele de întrerupere, iar 8052 de 6 nivele. Există 2 surse externe, INT0 şi INT1, care pot fi active pe nivel sau pe front în funcţie de biţii IT0 şi IT1 din registrul special TCON. Indicatoarele de întrerupere sunt biţii IE0 şi IE1 care se setează automat când apare o întrerupere şi se resetează când s-a încheiat tratarea întreruperii.

Întreruperile generate de timere sunt date de biţii TF0 şi TF1 din registrul TCON. Aceşti biţi sunt setaţi când apare o depăşire în numărătoarele/ temporizatoarele 0 şi 1. Întreruperea pentru timerul 2 va fi generată de un SAU între TF2 şi EXF2. Rutina de servire a întreruperii va determina care dintre aceşti biţi a cerut întrerupere prin citirea registrului T2CON.

Întreruperea de port serial este generată de un SAU logic între RI şi TI. . Rutina de servire a întreruperii va determina care dintre aceşti biţi a cerut întrerupere prin citirea registrului SCON.

Fiecare din sursele de întrerupere pot fi activate sau dezactivate prin setarea sau resetarea unui bit din registrul SFR numit IE, cu structura:

EA, X, ET2, ES, ET1, EX1, ET0, EX0, unde:
EA - dezactivează toate întreruperile cu IE=0. Cu IE=1 sunt validate întreruperile şi se pot masca individual;
ET2 - mascare întrerupere pentru timer-ul 2;
ES - mascare întrerupere port serial;
ET1 - mascare întrerupere timer 1;
EX1 - mascare întrerupere externă INT1;
ET0 - mascare întrerupere timer 0;
EX0 - mascare întrerupere externă INT0.

Întreruperilor pot fi tratate conform unei ierarhii implicite sau nivelul priorităţilor poate fi programat în registrul SFR numit IP.

priorităţi implicite:
IE0 (cea mai mare prioritate), TF0, IE1, TF1, RI+TI, TF2+EXF2.

structura registrului IP:
X, X, PT2, PS, PT1, PX1, PT0, PX0, unde:

PT2 - stabileşte nivelul de prioritate pentru întreruperea timerului 2;
PS - stabileşte nivelul de prioritate pentru întreruperea portului serial;
PT1 - stabileşte nivelul de prioritate pentru întreruperea timerului 1;
PX1 - stabileşte nivelul de prioritate pentru întreruperea externă 1;
PT0 - stabileşte nivelul de prioritate pentru întreruperea timerului 0;
PX0 - stabileşte nivelul de prioritate pentru întreruperea externă 0.

Nivelul priorității poate fi programat în HIGH sau LOW. O întrerupere LOW poate fi întreruptă de o întrerupere LOW dar nu poate fi întreruptă de o întrerupere HIGH. O întrerupere HIGH nu poate fi întreruptă.

4.1.6 Operarea cu economie de energie

8051 are două moduri de operare cu putere redusă; modul inactiv (Idle) şi modul cu tensiune scăzută (Power Down).

În modul inactiv oscilatorulfuncționează, dar nu se execută nici o instrucțiune. Timerele şi portul serial funcționează şi orice întrerupere de la ele readuc circuitul în stare normală. Un RESET hardware readuce circuitul în stare normală. Starea CPU se păstrează în întregime: indicator de stivă, Program Counter, registre etc.

În modul cu tensiune scăzută oscilatorul intern este oprit şi toate funcțiile sunt oprite. Se păstrează doar RAM-ul intern şi registrele speciale. Singura modalitate de a ieşi din această stare este prin RESET hardware. În această stare tensiunea de +5V poate fi redusă.

Modurile de operare cu economie de energie pot fi comandate prin registrul SFR PCON, care are structura: PD, IDL unde:

PD - bit pentru modul cu tensiune scăzută;
IDL - bit pentru modul inactiv.

4.1.7 Formarea unor semnale externe

RESET-ul se comandă pe intrarea RST a MC. Conținutul registrelor speciale SFR va fi adus la 00H, cu excepția SBUF care va fi nedeterminat şi PCON care va fi 0XXX0000 în binar. RAM-ul intern nu este afectat. La pornirea sistemului se poate realiza un RESET automat cu un montaj ca în figura 4.2.

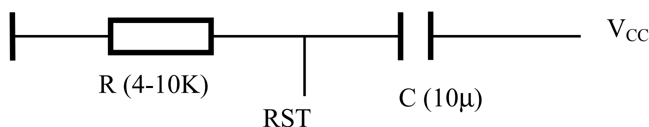


Figura 4.2 Circuit pentru formarea semnalului RESET

Ca şi generator de tact se poate folosi oscilatorul intern la intrările X1 şi X2, ca în figura 4.3.

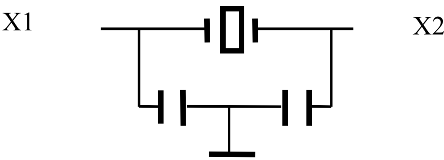


Figura 4.3 Circuit pentru folosirea generatorului intern de tact

Circuitul poate fi folosit şi cu un tact extern, legând X2 la masă, iar la X1 se leagă un oscilator extern.

4.1.8 Programarea EPROM-ului intern

Pentru a fi programat circuitul trebuie să fie alimentat şi să aibă cuplat generatorul de tact pentru că transferul de date se face prin bus-ul intern al circuitului. Adresa pentru EPROM-ul de programat trebuie să fie aplicată la portul 1 şi pinii P2.0-P2.3 ai portului 2, iar octetul de date de programat se aplică la portul 0. Ceilalţi pini ai portului 2, precum şi semnalele RST, /PSEN, /EA/Vpp trebuie să aibă următoarele nivele (tabelul 4.6).

Tabeul 4.6

Condiţii pentru operaţiile de programare, verificare, setare bit siguranţă

Operaţia	RST	nPSEN	ALE	**/EA/VPP**	P2.7	P2.6
Programare	1	0	impuls la 0 pentru 50ms	Vpp	1	0
Verificare	1	0	1	1	0	0
Setarea bitului de siguranţă	1	0	impuls la 0 pentru 50ms	VPP	1	1

Semnalul ALE este un impuls la 0 pentru 50 ms pentru a efectua programarea. Tensiunea Vpp este de +21V pentru 8751H şi de -12,75V pentru 8752BH **(A se consulta cu atenţie foile de catalog).** Sursa de tensiune pentru Vpp trebuie să fie foarte bine filtrată şi stabilizată deoarece chiar şi un mic impuls de tensiune poate produce vătămarea circuitului.

Verificarea programului se poate face dacă bitul de siguranţă nu a fost programat. Citirea se face în aceleaşi condiţii ca şi scrierea, cu excepţia lui ALE care este 1, /EA este 1, iar strobul de citire este P2.7. Pentru a nu se putea citi

neautorizat conținutul EPROM-ului se programează un aşa numit bit de siguranță care odată programat nu mai permite nici un acces electric la memoria EPROM. Odată programat, acest bit se poate şterge doar prin ştergerea întregii memorii EPROM.

Ştergerea EPROM-ului se poate face la lumină ultravioletă (2537Angstrom), cu cel puțin 15W.sec/cm^2, adică o expunere de 20-30 min. la distanța de 2-3cm la o lampă de ultraviolete cu 12W/cm^2.

Programarea rapidă (Quick Pulse Programming) poate fi aplicată la circuitele 875XBH, şi are ca rezultat posibilitatea programării unui circuit în numai 25 secunde. Programarea se poate face cu o tensiune Vpp mai mică (12,75V). Semnalul ALE are forma unor impulsuri multiple, 25 de impulsuri de 100 μs.

4.2 PROGRAMAREA MC DIN FAMILIA MCS-51

8051 are 111 instrucțiuni din care 64 sunt executate într-un singur ciclu.

4.2.1 Setul de instrucțiuni 8051

Instrucțiuni pentru transferul de date:

Transferuri generale:

MOV - efectuează un transfer pe bit sau pe octet de la sursă la destinație.

PUSH - incrementează registrul SP (Stack Pointer) şi transferă un octet de la sursă la locația din stivă adresată de SP.

POP - transferă un operand pe un octet de la locația din stivă adresată de SP la destinație şi decrementează SP.

Transferuri specifice acumulatorului:

XCH - mută octetul din sursă în acumulator.

XCHD - mută biții LOW din octetul din sursă în acumulator.

MOVX - mută un octet între memoria externă de date şi acumulator. Adresa memoriei de date externă poate fi specificată în registrul dublu DPTR.

MOVC - se citeşte un octet din memoria externă de program în acumulator.

Transferuri de adrese:

MOV DPTR, #data, încarcă imediat 16 biți de date în registrul dublu DPTR.

Instrucțiuni aritmetice:

Adunare:

INC - adună operandul cu 1 şi pune rezultatul ca nou operand.
ADD - adună acumulatorul cu operandul sursă şi pune rezultatul în A.
ADDC - adunare ca la ADD dar se adună şi bitul CY (Carry) din PSW.
DAA - ajustare zecimală, corectează suma care rezultă din adunarea a doi operanzi zecimali cu maxim 2 cifre.

Scădere:

SUBB - scădere cu împrumut, scade operandul din acumulator, apoi scade CY şi pune rezultatul în A.
DEC - scade 1 din operand şi pune rezultatul ca nou operand.

Înmulţire:

MUL - execută o înmulţire fără semn între acumulator şi registrul B, rezultatul obţinut fiind pe 2 octeţi (octetul cel mai semnificativ se pune în B). Dacă toţi biţii din octetul cel mai semnificativ al rezultatului sunt 0 bitul OV şi CY din PSW se pun la 0.

Împărţire:

DIV - execută împărţirea fără semn a conţinutului registrului A la conţinutul registrului B. Partea întreagă a rezultatului se pune în A, iar partea fracţionară în B.

Instrucţiuni pentru operaţii logice:

Instrucţiuni logice cu un singur operand:

CLR - resetează A orice bit adresabil direct.
SETB - setează orice bit adresabil direct.
CPL - complementează conţinutul lui A, fără a afecta PSW.
RL - rotaţie stânga a acumulatorului.
RLC - rotaţie stânga a A prin CY.
RR - rotaţie dreapta a A.
RRC - rotaţie dreapta a A prin CY.
SWAP - inversează niblurile în A.

Instrucţiuni logice cu 2 operanzi:

ANL - ŞI logic.
ORL - SAU logic.
XRL - SAU EXCLUSIV logic.

Instrucţiuni de control:

Apeluri şi salturi necondiţionate:

ACAL - este o instrucţiune pe 2 octeţi de apelare a unui subprogram care se foloseşte atunci când adresa de salt este cuprinsă în 2K ai paginii curente. Câmpul de adresă de 11 biţi este concatenat cu cei mai semnificativi 5 biţi din PC.
LCALL- este o instrucţiune pe trei octeţi de apelare a unui subprogram care adresează toţi cei 64K ai memoriei.
RET - transferă controlul la adresa de întoarcere care a fost în prealabil salvată în stivă şi decrementează registrul SP cu 2.

AJMP - este un salt necondiționat la adresa specificatăanalog cu ACALL.
LJMP - este un salt necondiționat la adresa specificată analog cu LCALL.
SJMP - este un salt necondiționat scurt în cadrul a 256 de octeți.

Salturi condiționate:

JZ - execută salt dacă acumulatorul este 0.
JNZ - execută salt dacă acumulatorul nu este 0.
JC - execută salt dacă bitul de CY (Carry) este 1.
JNC - execută salt dacă bitul de CY este 0.
JB - execută salt dacă bitul adresat este 1.
JNB - execută salt dacă bitul adresat este 0.
JBC - execută salt dacă bitul adresat este 1 şi apoi şterge bitul adresat.
CJNE - compară primul operand cu al doilea operand şi face salt dacă aceştia nu sunt egali.
DJNZ - decrementează operandul sursă şi pune rezultatul în operandul destinație. Dacă rezultatul nu este 0 se execută salt.

Întreruperi:

RETI - ca şi RET, dar activează întreruperile.

4.2.2 Modurile de adresare

Adresare prin registre - programatorul are acces la 8 registre de lucru, notate R0-R7. Cei mai puțin semnificativi 3 biți ai codului instrucțiunii indică unul dintre aceste registre. Se poate forma astfel o instrucțiune de un singur octet. De exemplu adunarea registrului R0 cu R1, cu rezultatul în acumulator:

```
MOV A,R0
ADD A,R1
```

Adresare directă - se pot adresa locații din RAM, porturi I/O sau registrele cu funcții speciale. La codul instrucțiunii se adaugă un octet care reprezintă locația care se foloseşte. De exemplu se adună conținutul locației 30 din RAM la conținutul locației 40, cu rezultatul în locația 40:

```
MOV A,30h
ADD A,40h
MOV 40h,A
```

Adresare indirectă prin registre - introdusă pentru a putea lucra cu variabile al căror loc în RAM se modifică în cursul rulării programelor. Ca registre index se folosesc registrele R0 şi R1, al căror conținut indică adresa în RAM. Cel mai puțin semnificativ bit al codului instrucțiunii indică registrul care este folosit ca index. În limbajul de asamblare al lui 8051, adresarea indirectă se reprezintă cu @. De exemplu se adună conținutul locației adresată de registrul R0 cu conținutul locației adresată de registrul R1, cu rezultatul în acumulator:

```
MOV A,@R0
```

ADD A,@R1

Adresare imediată - folosită când operandul este o constantă cu o valoare cunoscută, care se specifică în codului instrucțiunii. În limbajul de asamblare al lui 8051 constanta este precedată de semnul #. De exemplu adunarea lui 15 cu 18 zecimal, cu rezultatul în acumulator:

MOV A,#15
ADD A,#18

4.3 ECHIPĂRI SPECIALE CU MEMORIE

4.1.1 Memoria EEPROM (Philips 80C851)

Memoria EEPROM are dimensiunea de 256 octeți, poate reține informațiile minimum 10 ani și poate fi supusă la 10.000 de cicluri de ștergere/scriere. Circuitul conține multiplicatorul de tensiune pentru ștergere și scriere.

Comunicația între UC și EEPROM se realizează cu ajutorul a 5 registre:

EADRH (adresa F3h), **EADRL** (adresa F2h) sunt două registre pentru adresare, primul pentru partea LOW a adresei, celălalt pentru partea HIGH (pentru implementări viitoare și pentru adresarea biților de securitate).

EDAT (adresa F4h) este registrul de date în care se stochează octetul de scris sau octetul citit. Sunt posibile și ștergeri pe bloc de date, caz în care conținutul acestui registru nu contează.

ETIM (adresa F5h) este un registru pentru timer necesar pentru a adapta timpul de citire/ scriere la frecvența sistemului și trebuie încărcat cu valori funcție de tactul sistemului și de caracteristicile EEPROM-ului.

ECNTRL (adresa F6h) este registrul de control care:

- stabilește modurile de lucru: scriere, citire, ștergere pe octet, ștergere pe bloc;
- conține un bit care semnalează că este în curs o scriere sau ștergere.

Schema bloc a modulului este dată în figura 4.4.

Secvențiatorul asigură secvența de timp corespunzătoare pentru scriere sau ștergere. Datele și adresele se transferă prin intermediul registrelor de pe magistrală.

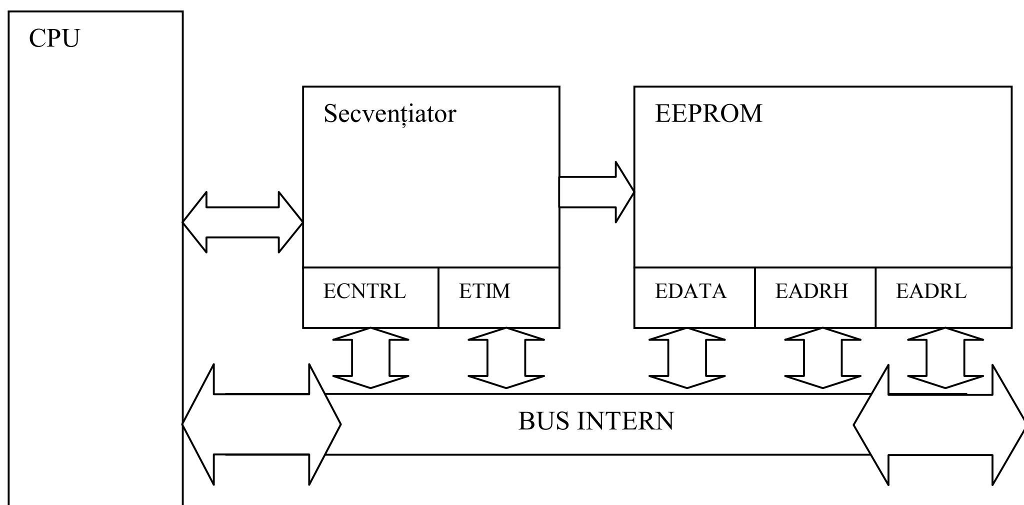


Figura 4.4 Schema bloc a modulului EEPROM

O citire se poate realiza simplu:

```
Citire:  MOV   EADRL,#20H
         MOV   A,EDAT
```

Ca urmare a acestei secvențe de program, conținutul locației 20h este citit în acumulator.

Memoria EEPROM este protejată cu un octet la adresa 8000h. Se poate valida securitatea cu următoarea secvență de program, cu scrierea activată:

```
Activare securitate:
         MOV   EADRH,#80H
         MOV   EADRL,#00H
         MOV   EDAT,#FFH
```

Acest octet nu mai poate fi modificat prin soft. Programul din EEPROM nu mai poate fi citit sau modificat cu instrucțiuni MOVC din memorii externe, ci doar executat.

4.3.2 Memoria FLASH cu programare paralelă (Atmel AT89C55)

Multe MC compatibile 8051 sunt echipate cu memorie FLASH. Unul dintre acestea este AT89C55 care este echipat cu 20K octeți memorie FLASH care

poate fi programată şi ştearsă (EPROM) în maximum 1000 de cicluri de scriere/ştergere.

Programarea memoriei FLASH se poate face cu o tensiune mare, de +12V aşa încât se pot folosi inscriptoarele de EPROM sau se poate face cu +5V, pentru ca programarea să fie posibilă în sistemul gazdă. AT89C55 este fabricat cu memoria ştearsă (plină cu FFh) şi gata de a fi programată.

Programarea memoriei se face astfel:

- pe liniile de adresă se stabileşte adresa locației de programat;
- pe liniile de date se stabileşte octetul de înscris;
- se aplică un front pe /EA/Vpp la +12V (pentru programarea cu 12V);
- se aplică un impuls ALE//PROG.

Starea programării este indicată de bitul RDY//BSY (P3.4), linia fiind LOW în timpul programării şi HIGH când programarea s-a terminat.

După înscriere se poate face verificarea a ceea ce s-a înscris, prin adresare şi citirea octetului de date.

Toată memoria FLASH se poate şterge electric aplicând semnalele de comandă corespunzătoare (din tabelul care există în foile de catalog) şi aplicând apoi un impuls ALE//PROG de 10ms. Asemănător se programează (programare paralelă) şi circuitul 89C51 de la Philips.

4.3.3 Memoria FLASH cu programare paralelă şi serială ISP (Philips 89C51RC)

O facilitate interesantă şi utilă o au MC care au înscris în ROM un mic program monitor care poate să gestioneze înscrierea memoriei FLASH prin canalul serial. Modul de înscriere serial se numeşte *In-System Programming* (ISP) şi este realizat printr-un canal serial cu liniile TxD şi RxD şi liniile de alimentare de +5V şi masă, precum şi tensiunea necesară înscrierii memoriei FLASH, +Vpp. Softul care gestionează canalul serial este un monitor înscris în ROM. După programarea memoriei FLASH, ROM-ul poate fi invalidat. Programul monitor determină rata de transfer cu care i se trimit date şi transmite în ecou ce a recepționat. După transmiterea caracterului pentru stabilirea ratei de transfer, se transmite un octet de identificare care stabileşte natura datelor care urmează. Numărul de octeți care urmează este limitat la 16. În foile de catalog sunt explicate comenzile care pot fi date pe această cale. Programul monitor ocupă 1K şi ROM-ul se numeşte Boot ROM.

La această memorie FLASH timpul de acces este de 100ns, timpul necesar înscrierii unei locații este de 20ms, iar ştergerea se realizează în 3 secunde.

4.4 INTERFEȚE ŞI PERIFERICE ON CHIP SPECIALE

4.4.1 Convertorul A/D

Circuitul analogic de intrare constă într-un multiplexor analogic şi un convertor A/D de 8 biți cu aproximații succesive. Tensiunea de referință pentru convertor şi masa analogică sunt conectate prin pini speciali. O conversie poate avea loc în 24 sau 48 de cicli maşină, programabil, ceea ce înseamnă un timp de conversie de 24μs la un tact de 12MHz.

Convertorul este controlat de registrul de control ADCON care selectează şi canalul de conversie. Terminarea conversiei este semnalizată cu un bit tot în ADCON, iar rezultatul conversiei este stocat în registrul ADCH. O conversie poate fi declanşată în 3 feluri:

- start în operare normală şi revenire în operare normală;
- start în operare normală revenire în mod inactiv (Idle);
- intrare în mod inactiv şi declanşarea unei conversii din exterior prin pinul STDAC.

Cu registrul ADCON (C4h) se poate programa:

- selecția canalului analogic dorit;
- se poate programa ca o conversie să fie declanşată de pinul extern STADC;
- se poate declanşa o conversie;
- conține un bit care semnalează că s-a terminat conversia. Cu acest bit se poate solicita o cerere de întrerupere;
- se poate selecta viteza de conversie la viteza maximă (24 cicli) sau mai mică (48 de cicli).

4.4.2 Interfața PWM

Circuitul este prevăzut cu un canal PWM la care frecvența de repetiție este programată cu un registru de prescalare (PWMP- adresa FEh) care generează un ceas pentru un numărător de 8 biți. Conținutul numărătorului este comparat cu cel al registrului PWM0 (adresa FCh); dacă numărul este mai mare ieşirea /PWM0 este LOW, dacă este mai mic sau egal /PWM0 este HIGH. Factorul de umplere poate fi astfel modificat între 1/255 şi 255/255.

4.1.3 Interfața I²C (Siemens P80CL580)

Portul serial I²C are 2 linii, date seriale (SDA) pe poziția liniei P1.7 şi ceas serial (SCK) pe poziția bitului P1.6. Interfața lucrează în 4 moduri:

- transmițător MASTER
- receptor MASTER
- transmițător SLAVE
- receptor SLAVE

Aceste funcții pot fi controlate de registrul S1CON (Serial Control Register) şi S1STA (Serial Status Register). Cu datele se lucrează prin S1DAT (Data Shift Register) iar adresa se stabileşte în S1ADR (Slave Address Register), figura 4.5.

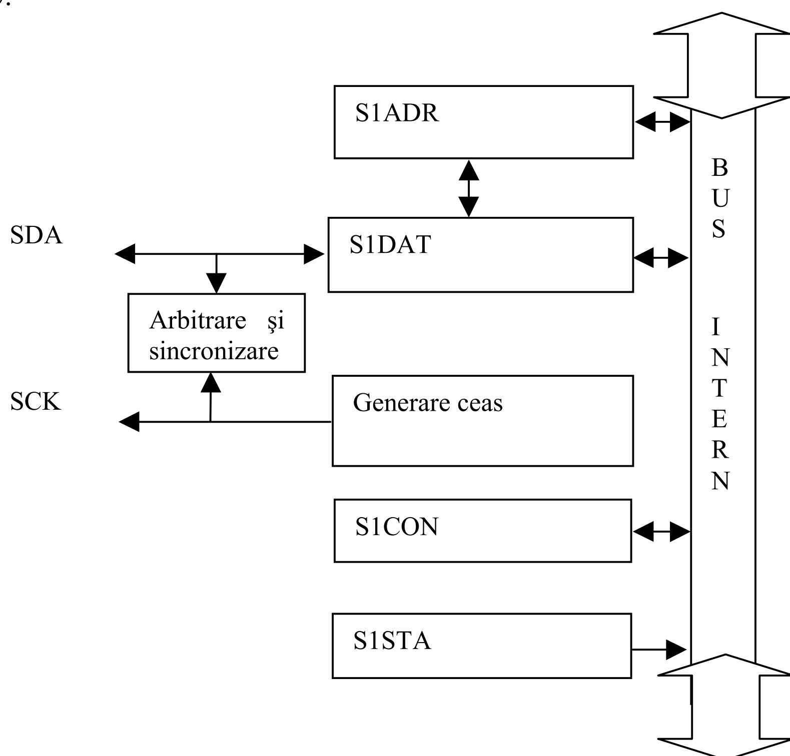


Figura 4.5 Interfață I²C

Cu registrul de control S1CON (registru SFR la adresa DBh) se pot programa:

- ceasul serial de transfer în mod MASTER (SCK), care este în funcție de tactul sistemului şi poate fi de maximum 100kHz;
- se poate selecta dacă liniile I/O (P1.6 şi P1.7) au semnificațiile generale sau speciale pentru I²C;

- se poate porni transferul prin generarea de condiţii de START repetate (mod MASTER) sau verificarea bus-ului şi generarea de START doar dacă bus-ul este liber (mod SLAVE);
- se poate opri transferul prin generarea unei condiţii de STOP;
- se pot valida întreruperile care se generează în următoarele condiţii: s-a generat o condiţie de START, s-a recepţionat adresa proprie, un octet s-a transmis sau s-a recepţionat;
- se poate insera un ACK, (nivel LOW pe SDA) după recepţia unui caracter sau a adresei proprii.

Registrul de stare S1STA (D9h - registru Read Only) poate fi folosit pentru generarea unei întreruperi şi saltul la o rutină de servire.

În registrul de date S1DAT (DAh) se înscrie octetul care se transmite sau se recepţionează; cel mai semnificativ bit se transmite sau se recepţionează primul.

În registrul de adrese S1ADR (DBh), la un dispozitiv MASTER se stabileşte adresa dispozitivului SLAVE cu care doreşte un transfer de date.

Observaţie: convertorul AD, interfaţa I^2C şi canalul PWM0 pot lucra în modurile cu economie de energie. Convertorul, interfaţa I^2C şi canalul PWM0 rămân active în modul Idle al UC şi pot genera o întrerupere sau un RESET, terminând astfel modul inactiv al UC.

4.4.4 **Interfaţa USB** (EZ-USB seria 2100)

Familia EZ-USB de la Anchor Chips (www.anchorchips.com) echipează MC-ul lor echivalent cu 8051 cu un modul USB inteligent, destinat legăturii USB de mare viteză (12Mbps). Modulul USB inteligent admite instrucţiuni avansate, de aceea punerea la punct a lucrului cu USB devine mai rapidă. MC este echipat cu RAM care poate fi încărcată de la un PC. Din acest motiv circuitul nu mai are ROM. Circuitul mai conţine şi o interfaţă I2C, precum şi linii I/O de uz general. Tot ca un avantaj se poate menţiona că bus-ul de date şi adrese nemultiplexat este accesibil la pini speciali, ceea ce înseamnă că nu se sacrifică pini I/O pentru cuplarea unor componente exterioare şi nici nu este nevoie de latch-uri pentru separarea datelor de adrese.

Schema bloc a acestui MC este dată în figura 4.6.

Modulul USB realizează în timpul iniţializării o enumerare şi alocare de adrese a dispozitivelor USB conectate. Această operaţie este posibilă ca urmare a mutării unei secvenţe de program din RAM-ul MC în RAM-ul modulului USB.

Încărcarea programului în RAM se poate face atât de la un sistem PC cât şi de la un EEPROM serial prin interfaţa I2C sau clasic, prin conectarea unei memorii ROM externe. Operaţia de enumerare iniţială permite identificarea unui corespondent USB şi creează posibilitatea încărcării programelor de la sistemul gazdă chiar prin USB.

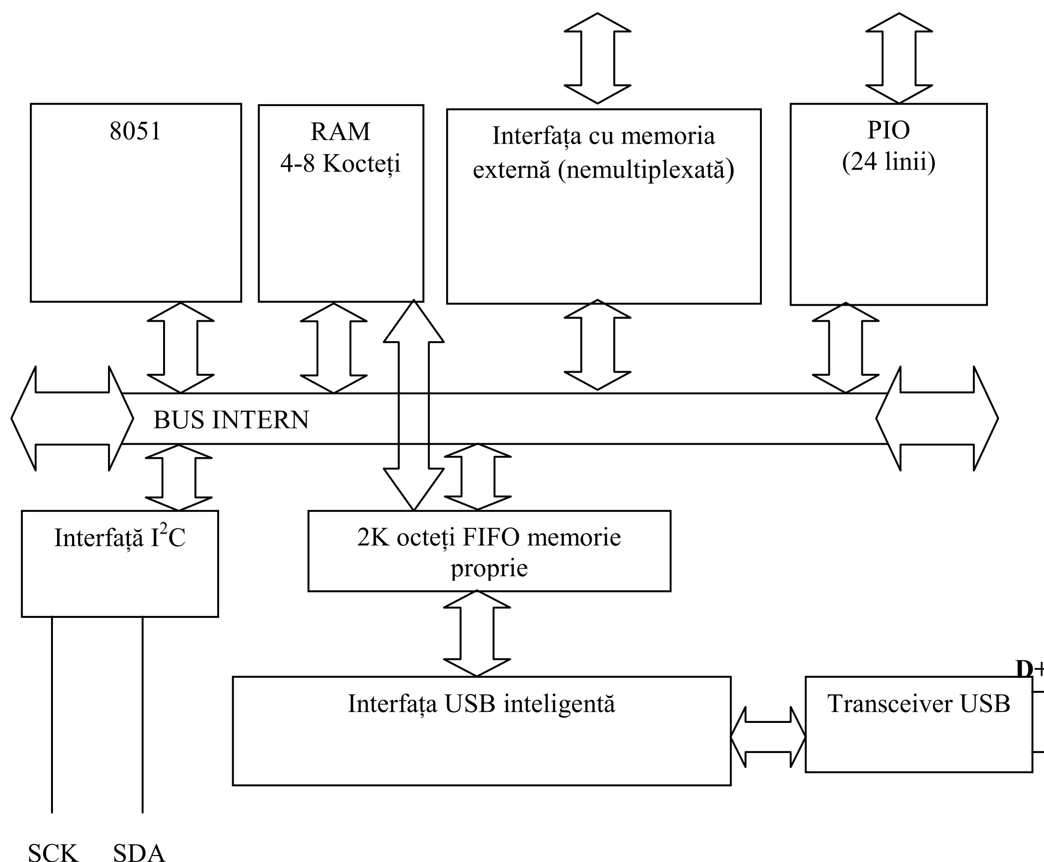


Figura 4.6 Schema bloc a unui MC cu interfață USB

4.4.5 Aria de numărătoare programabilă (PCA)

Aria de numărătoare programabilă este un circuit special de timp format din 5 module de 16 biți cu posibilitatea de captură şi comparație care se adaugă timerelor obişnuite ale MC. Fiecare modul poate fi programat individual să lucreze în unul din modurile:

- captură pe front pozitiv sau negativ;
- timer;
- canal PWM;
- ceas de gardă (doar modulul 4).

Fiecare timer are un pin asociat din portul 1, ca în figura 4.7.

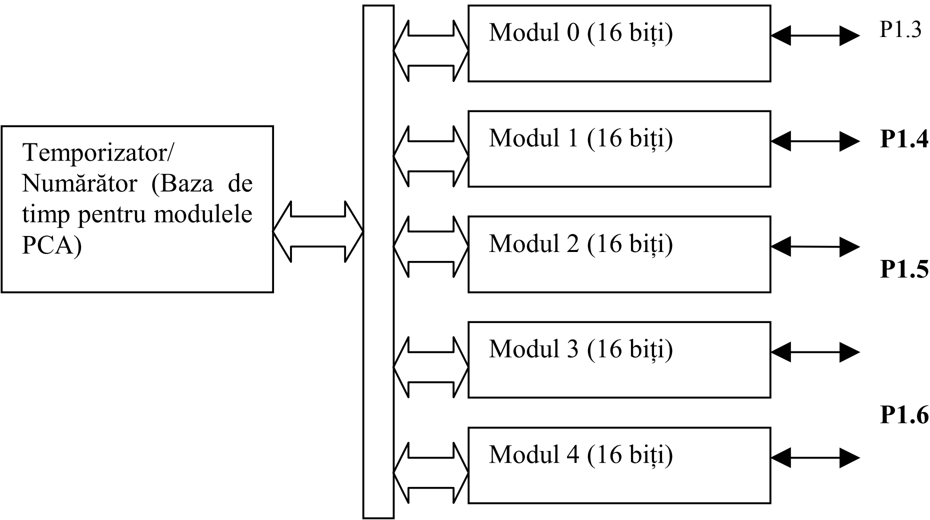


Figura 4.7 Structura ariei de timp programabile (PCA)

Timerul comun pentru toate modulele este un timer obişnuit. El poate funcționa cu diferite tacte programate în registrul SFR CMOD cu 2 biți, conform tabelului 4.7.

Tabelul 4.7

Programarea tactului pentru timer

CPS1	CPS0	Tact pentru timer
0	0	frecvența oscilatorului/12
0	1	frecvența oscilatorului/4
1	0	Semnalul de depăşire de la timerul 0 standard
1	1	De la un pin extern (ECI, P1.2)

Fiecare modul are asociat un registru de comandă (CCAPM0-CCAPM4) care controlează modul de operare al modulului:

- se poate valida ca o coincidență în modul să genereze o întrerupere;
- se poate valida modul PWM;
- se poate valida ca ieşirea P1.x să schimbe starea dacă a apărut o coincidență între conținutul modulului şi conținutul timerului;
- se poate programa pe care front al intrării P1.x să se facă numărarea impulsurilor externe.

Fiecare modul mai are asociat un registru de 16 biți (câte două de 8 biți; CCAP0H-CCAP5H şi CCAP0L-CCAP5L) care stochează valoarea numărată la apariția unei coincidențe. În mod PWM aceste registre controlează factorul de umplere.

Funcționarea PCA:

- în modul de captură, când apare o tranziție pe intrarea externă P1.x, se încarcă valoarea la care a ajuns timerul comun în registrele de date (CCAPxH şi CCAPxL). În acest moment se poate genera o întrerupere;
- în mod timer, conținutul registrelor de date este incrementat de la intrarea externă. Când se ajunge la o valoare egală cu cea stocată în timerul comun poate fi generată o întrerupere;
- în mod PWM fiecare modul poate fi folosit ca un canal independent. Frecvența semnalului PWM este aceeaşi şi depinde de sursa timerului comun. Factorul de umplere se poate modifica prin registrul CCAPxL;
- în mod ceas de gardă utilizatorul încarcă registrul CCAPxH şi CCAPxL. Când timerul comun ajunge la o valoare egală cu cea stocată de utilizator se generează un RESET intern. Pentru ca să nu se ajungă la RESET într-un progam rulat normal, utilizatorul trebuie periodic să schimbe valoarea din timer sau să reseteze ceasul de gardă.

4.4.6 MC cu interfață pentru RAM nevolatil - NVRAM (Dallas DS5000FP)

Un astfel de MC poate adresa o memorie externă SRAM (între 8K şi 64K) care poate fi făcută nevolatilă prin alimentarea cu baterii. O baterie cu litiu poate funcționa cca. 10 ani. Transferul de date cu memoria NVRAM se face pe un bus separat pentru a nu micşora numărul de linii I/O.

Circuitul nu are ROM pentru programul utilizator, programul fiind stocat în NVRAM, programarea se realizează în sistem, prin interfața serială a MC. La acest tip de MC programul se poate schimba chiar şi în timpul funcționării. Programul se poate încărca inițial prin interfața serială, sub comanda unui program existent într-un ROM intern numit Boot ROM (sau BOOTSTRAP LOADER ROM) care este apoi invalidat şi devine invizibil la adresare.

4.4.7 Interfața LCD (PHILIPS P83C434)

În jurul unui nucleu 8051 a fost construit un MC specializat pentru comanda panourilor LCD. Rămân disponibile pentru uz general 12 linii I/O. Modulul LCD are 24 de linii pentru comanda segmentelor, din care 2 pot fi folosite pentru comanda planurilor din spate. Afişajul poate fi comandat cu tensiuni variabile obținute intern prin divizarea tensiunii de alimentare cu rezistențe. Schema bloc a MC este dată în figura 4.8.

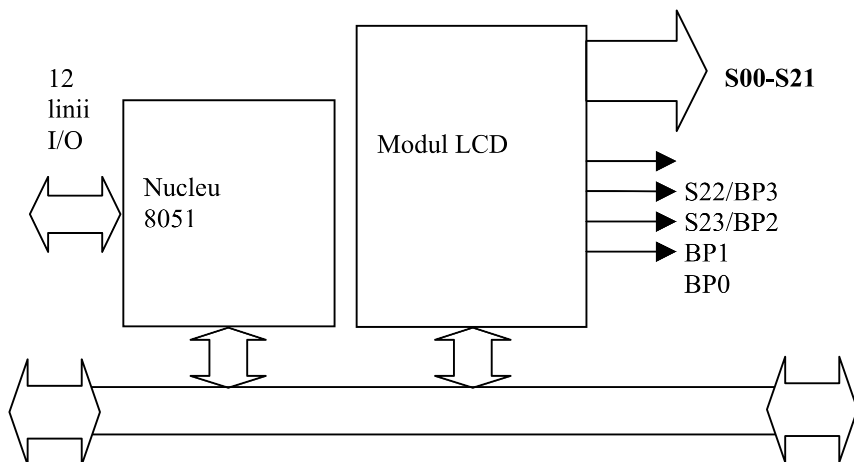


Figura 4.8 MC cu modul de comandă LCD

MC se poate folosi la comanda afişoarelor cu până la 4 planuri în spate. Cele 24 de linii de comandă a segmentelor pot comanda 12 caractere numerice formate cu 7 segmente sau 88 de elemente grafice. Funcţionarea afişajului poate fi mai bine înţeleasă în cazul concret al unui singur plan în spate, pentru 2 elemente alăturate ale afişajului, figura 4.9.

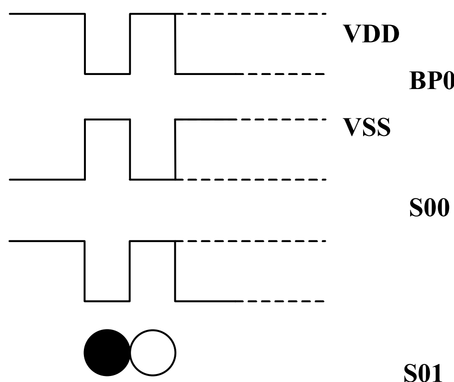


Figura 4.9 Comanda a doua elemente alăturate ale afişajului LCD

Primul element este aprins pentru că între segment şi planul din spate este o diferenţă de potenţial, iar al doilea este stins pentru că nu există o diferenţă de potenţial.

Comanda modulului de afişare se face cu 12 registre LCD0-LCD11 (adrese 9Ah-BFh) care conţin configuraţia segmentelor stinse/aprinse pentru fiecare plan din spate.

4.4.8 MC specializat pentru TV şi video (PHILIPS 83C145)

Acest MC este construit în jurul unui nucleu 8051 şi are 8-16Kocteți ROM sau OTP, 256 octeți RAM, controller pentru vizualizare pe ecran (On Screen Display OSD), 3 ieşiri video digitale, memorie RAM pentru display de 128x10 biți, generator de caractere (ROM 60 caractere x 18 linii x 14 puncte), 8 canale PWM de 6 biți şi un canal PWM de precizie de 14 biți, convertor numeric analogic. Nu se poate conecta memorie externă. Schema bloc este dată în figura 4.10.

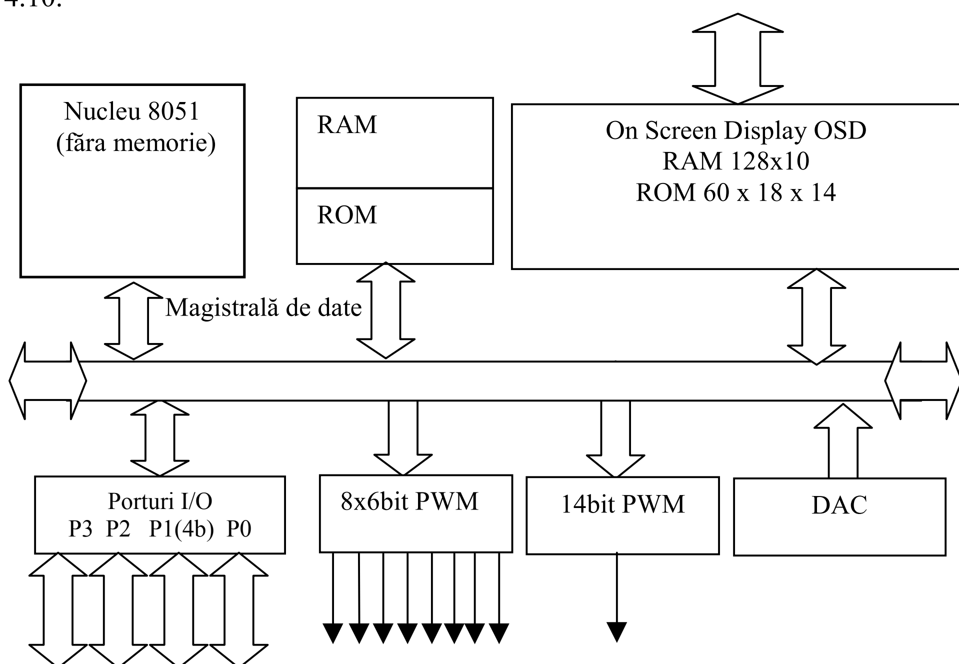


Figura 4.10 Schema bloc a unui MC specializat TV/video

Ceasul canalelor PWM se formează din tactul sistemului divizat cu 4. Acest tact este aplicat tuturor canalelor PWM şi unui numărător de 14 biți. Canalele de 6 biți utilizează doar partea mai puțin semnificativă a numărătorului de 14 biți. Fiecare canal PWM are asociat un registru SFR. La egalitatea valorii acestui registru cu conținutul numărătorului, ieşirea PWM schimbă starea. Structura canalelor PWM este reprezentată în figura 4.11.

Convertorul numeric analogic este folosit pentru a realiza conversia analog numerică soft (figura 4.12). Circuitul are 3 intrări analogice care pot fi comutate pe rând la intrarea unui comparator de tensiune. La cealaltă intrare a comparatorului se aplică ieşirea convertorului numeric analogic. Când intrările sunt egale, valoarea aplicat convertorului numeric analogic este chiar valoarea numerică a semnalului analogic de intrare.

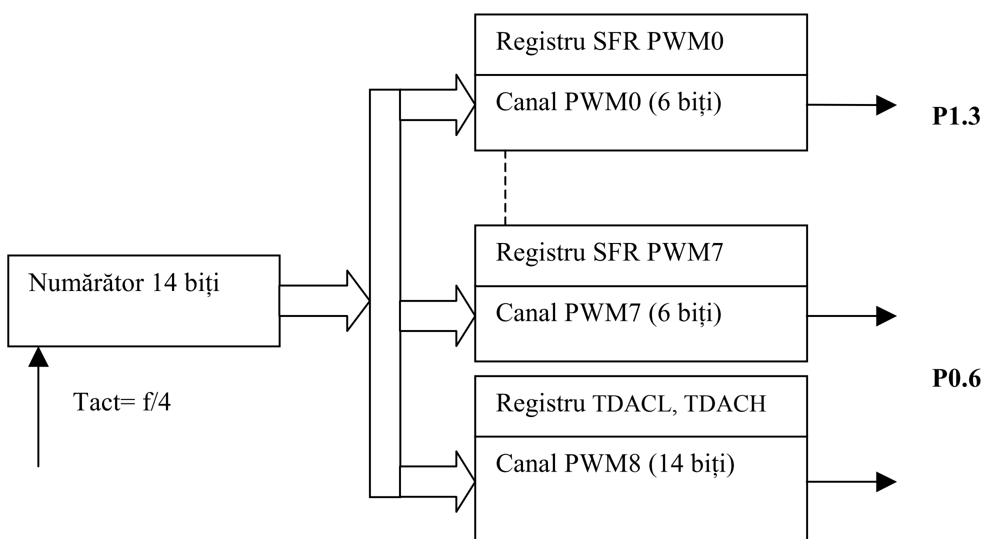


Fig. 4.11 Structura canalelor PWM (ale MC specializat TV/video)

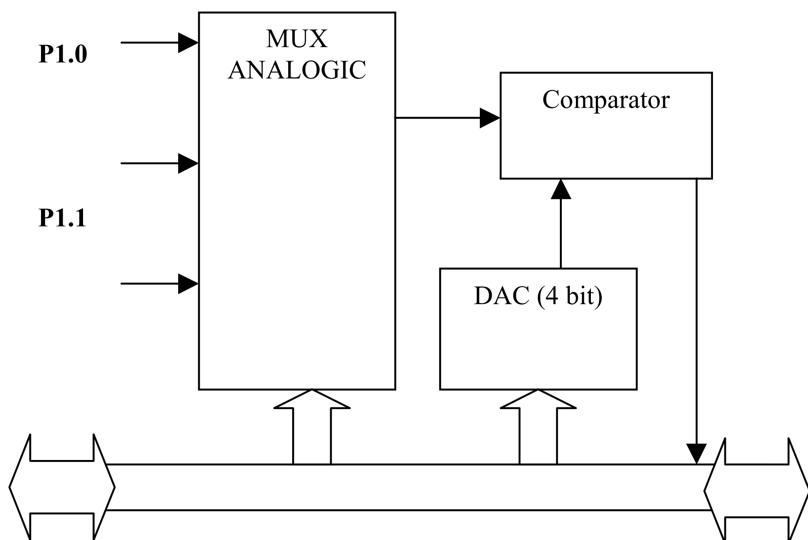


Figura 4.12 Convertorul analog numeric (al MC specializat TV/video)

Modulul OSD are rolul de a suprapune text pe o imagine de televiziune. Intrările în acest bloc sunt:

- 2 ceasuri video;
- semnalul de sincronizare orizontală;

- semnalul de sincronizare verticală.

Modulul OSD poate fi echipat cu EPROM, ceea ce înseamnă că generatorul de caractere poate fi programat. Modulul OSD permite 8 moduri de scriere umbrită a caracterului, culoarea caracterului este selectabilă, culoarea fondului este selectabilă.

4.4.9 **MC cu arie configurabilă** (TRISCEND E5)

Firma TRISCEND a realizat un MC compatibil 8051 care conține o arie de porți configurabilă (Configurable System-on-Chip CSOC). Acest MC integrează pe un singur chip un MC de uz general 8051, un bloc de RAM de mari dimensiuni şi o arie de module configurabile, toate acestea conectate între ele printr-un bus de mare viteză. Programarea ariei se face prin programul de inițializare şi se poate relua de ori câte ori.

Descrierea pe scurt a circuitului: rulează la o frecvență de 40MHz şi are echipare standard (256 octeți RAM, ceas de gardă, trei timere, 2 canale DMA, interfață serială UART, 64K octeți RAM) şi are în plus 3200 de module configurabile (CSL Configurable System Logic - ceea ce înseamnă cam 40.000 de porți).

Conectarea între modulele cofigurabile şi restul sistemului, respectiv liniile I/O se realizează prin magistrala CSI (Configurable System Interconnect), cu 8 biți de date bidirecțional, 32 de biți de adresă, rata de transfer de maxim 40MBps. Magistrala CSI permite lucrul multi master, master putând fi unitatea centrală 8051, canalele DMA sau interfața JTAG.

Interfața cu exteriorul poate fi realizată cu maxim 315 linii I/O (depinde de variantă şi capsulă), există şi posibilitatea conectării memoriei externe, iar un modul de interfață IEEE 1149.1 JTAG permite testarea sistemului. Liniile I/O au caracteristici programabile (curent de ieşire, histerezis la intrare etc.). Schema bloc este dată în figura 4.13.

Interfața cu memoria externă permite legarea directă a unei memorii de 256K x 8 biți (de regulă FLASH) pentru înscrierea inițială a programului în MC. Înscrierea inițială poate fi realizată la RESET din această memorie FLASH externă în mod paralel, dar poate fi realizată prin citirea programului prin interfața serială de la un PROM serial. Programul poate fi executat din FLASH sau poate fi copiat prin interfața serială în RAM şi executat de acolo.

Modul serial eliberează pini I/O care pot fi astfel folosiți în alte scopuri. Procedura de încărcare poate să nu fie reluată dacă RAM-ul se alimentează cu baterie.

Cu aria de module configurabile se pot realiza sisteme la cerere. Fiecare modul poate îndeplini diverse funcții, combinaționale sau secvențiale.

Modulele configurabile sunt aranjate într-o matrice, a cărei dimensiuni depind de varianta de circuit. Modulele sunt grupate câte două, pentru a putea împărți resursele. Schema bloc a unui nod din matrice este dată în figura 4.14.

8051
CPU
256 x 8 RAM
Ceas de gardă
3 x Timer
UART
Controller de întreruperi
2x DMA

RAM
64Kocteți

Interfața cu memoria externă

Interfața JTAG

BUS CSI

Matrice CSL

PIO PIO ------------------------ PIO

Figura 4.13 Schema bloc a unui MC TRISCEND E5

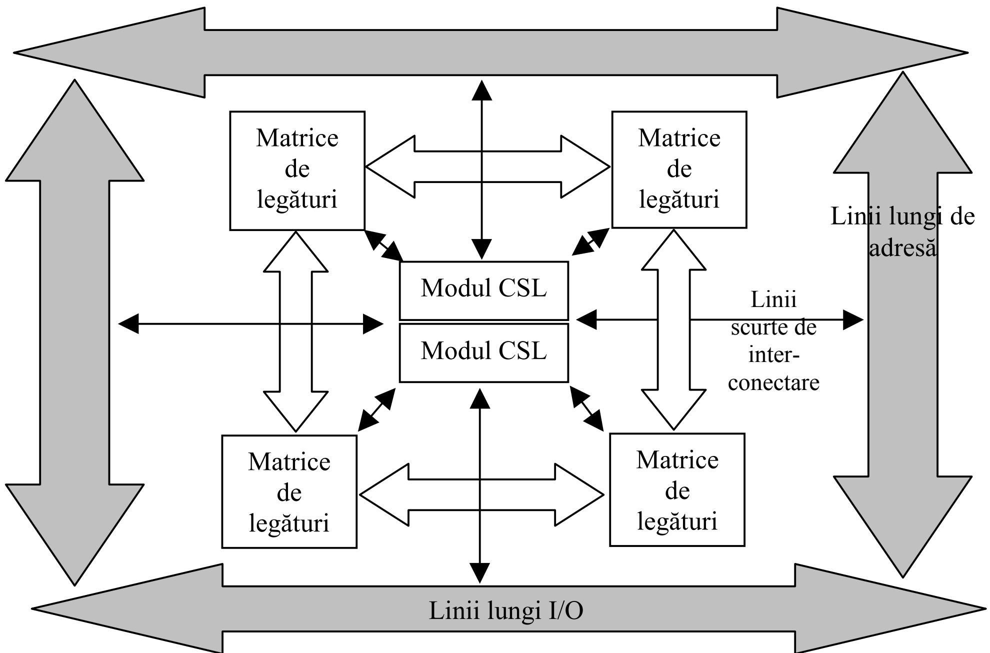


Figura 4.14 Schema bloc a unui modul de arie configurabilă

Structura unei celule este dată în figura 4.15.

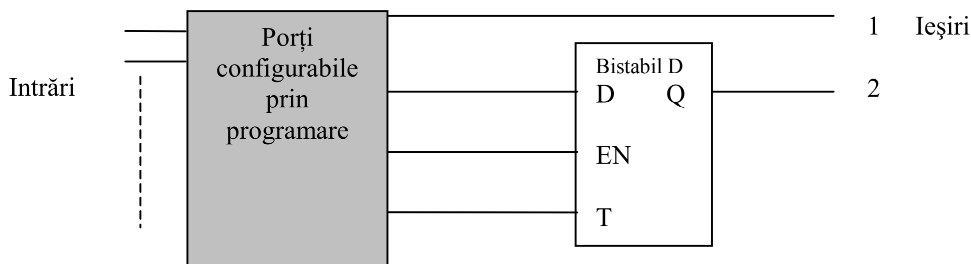


Figura 4.15 Structura unei celule configurabile

Structura porţilor poate fi programată la iniţializare şi reprogramată de ori câte ori. O celulă poate îndeplini una din funcţiile:

- logică (ieşirea 1);
- aritmetică;
- de memorare (ieşirea 2);
- de magistrală;
- secvenţială.

Testarea JTAG se poate realiza cu un calculator conectat la interfaţa JTAG. Pinii folosiţi sunt:

- TCK ceas de testare (intrare în MC);
- TMS comanda modului de test, intrare în MC, activ pe 0;
- TDI date seriale de intrare în MC;
- TDO date seriale de ieşire din MC.

În modul de testare nu este nevoie ca MC să aibă o memorie externă. Prin legătura JTAG se poate programa matricea CSL şi se poate observa modul de rulare al programului de către MC prin intercalarea de break point-uri, rularea pas cu pas, citirea registrelor interne etc.

4.5 SISTEM MINIMAL CU 8051

Schema electrică simplificată a unui sistem minimal cu 8051 este dată în figura 4.16.

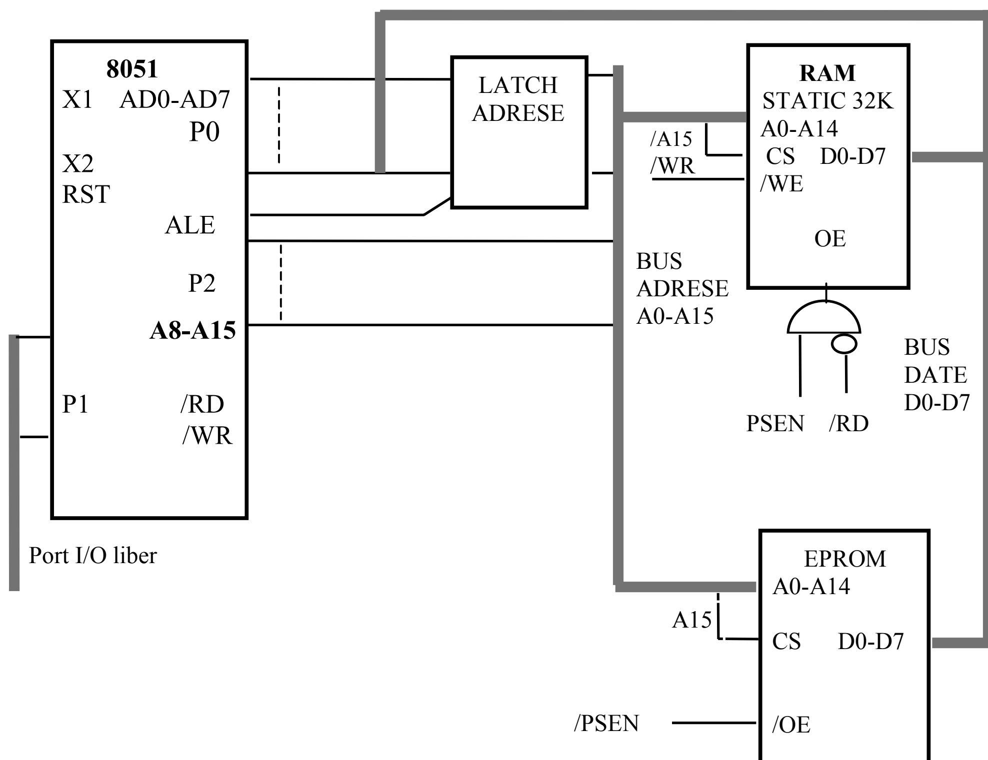


Figura 4.16 Sistem minimal cu 8051

Sistemul minimal conține RAM cu capacitatea de 32KB, selectat cu PSEN (care indică faptul că nu este acces la memoria program) şi /RD pentru o citire sau /WR pentru o scriere în RAM.

Bus-ul de date şi adrese este comun pentru AD0-AD7. Adresele se separă cu un latch validat de semnalul ALE. Liniile de date sunt validate cu /RD sau /WR.

Memoria program EPROM este opțională, (este necesară pentru modelele care nu au EPROM sau OTP intern, de exemplu 8031) fiind selectată cu semnalul PSEN. Dacă există EPROM extern /EA=LOW, iar dacă se foloseşte EPROM-ul intern /EA=HIGH.

Circuitul se completează cu un generator de tact şi două componente pentru realizarea RESET-ului, ca în unul din subcapitolele anterioare.

Pentru dezvoltări se poate folosi portul P1, care este liber, şi canalul serial. Dacă numărul liniilor I/O nu este suficient, se poate folosi un expander cum este de exemplu 8243, care poate extinde 4 linii la 4x4 linii bidirecționale (figura 4.17).

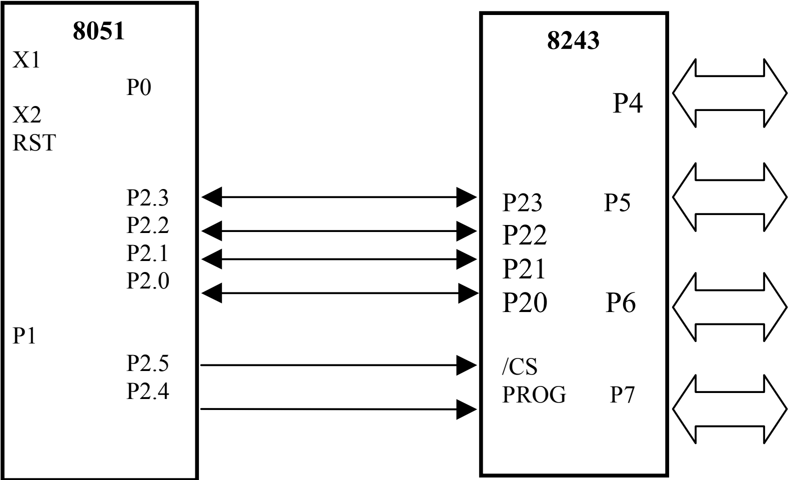


Figura 4.17 Expandarea liniilor I/O cu circuitul 8243

4.6 DATE COMPARATIVE PENTRU MC DIN FAMILIA MCS-51

Tabelul 4.8

Tabel comparativ pentru MC din familia MCS -51

MC	Magistrala de date	Frecvenț a (MHz)	Linii I/O	Interfețe speciale	Preț (USD)
PHILIPS P87C51	8	16	32	2	4
PHILIPS S87C552	8	16	48	5	35
PHILIPS XA-G3	16	30	32	2	90
TRISCEND E5	8	40	128	2	27
INTL 87C196	16	20	53	6	18

5

MICROCONTROLLER-E RISC

5.1 MICROCONTROLLER PIC

Începem prezentarea MC cu arhitectură RISC cu microcontrollerul PIC. Producătorul plasat cel mai bine pe piața MC PIC este Microchip (www.microchip.com). Un alt producător important de MC PIC este Motorola. Există mai multe familii de MC PIC; PIC12, PIC16, PIC17.

5.1.1 PIC12

MC PIC12 sunt MC cu prețuri mici şi uşor de utilizat datorită arhitecturii RISC. MC are un număr de 33 de instrucțiuni cu un grad mare de ortogonalitate, din care majoritatea se execută într-un singur ciclu, iar cele de salt în două cicluri. Spațiul mic ocupat (capsulă de 8 pini) fac aceste MC foarte potrivite aplicațiilor miniatură precum şi aplicațiilor casnice. Pentru a putea fi realizată o şi mai mare economie de spațiu, circuitul de RESET este integrat. Sunt disponibile circuite cu OTP, cu EPROM sau EEPROM inclus, care fac posibilă atât realizarea seriilor mari cât şi a prototipurilor sau a aplicațiilor cu elemente care se modifică (de exemplu aplicații de asigurare a securității cu coduri variabile). Aceste MC admit o frecvență de până la 4MHz.

MC PIC12 are o arhitectură Harvard, cu magistrale diferite pentru date şi pentru instrucțiuni. Acest lucru permite ca magistrala de instrucțiuni să fie mai mare (de 12 biți) şi ca urmare majoritatea instrucțiunilor pot fi de un cuvânt şi pot fi executate într-un singur ciclu.

Unitatea centrală este o unitate pe 8 biți care poate realiza funcții aritmetice şi booleene: adunare, scădere, deplasare şi operații logice cu date care se găsesc în registrul de lucru (W) şi în oricare registru de uz general. Setul de instrucțiuni are un mare grad de ortogonalitate, ceea ce reduce mult timpul necesitat de realizarea unei aplicații. O operație poate afecta biții de stare: Carry

(C), Digit Carry (C) şi Zero (Z). Numărătorul de program (PC) este un registru de 12 biţi, ceea ce înseamnă că poate adresa un spaţiu de 2K cuvinte de 12 biţi.

Schema bloc a acestui MC este dată în figura 5.1.

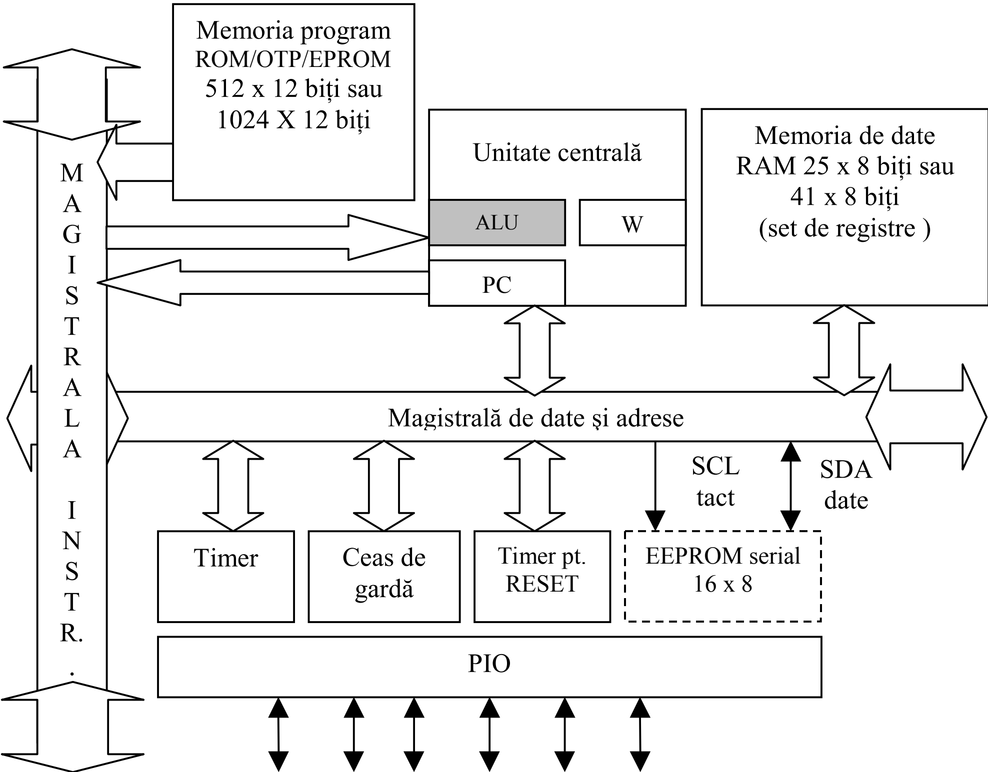


Figura 5.1 PIC12 –Schema bloc

Setul de registre localizat în RAM conţine registre cu funcţii speciale şi registre de uz general. Registrele generale pot fi apelate direct sau indirect, prin intermediul registrului special FSR (*File Select Register*). Registrele şi adresele lor sunt date în tabelul 5.1 (pentru circuitul PIC12C509).

Tabelul 5.1

Setul de registre PIC12C509

Registru	Adresa(H)	Registru	Adresa(H)
INDF	00	FSR	04
TMR0	01	OSCCAL	05
PCL	02	GPIO	06
STATUS	03	Registre generale	07-1F

Stiva este de 12 biţi şi este realizată hard. Nu există indicator de stivă şi nu sunt instrucţiuni de PUSH şi POP, lucrul cu stiva fiind automat la instrucţiunile CALL şi RETLW. Stiva admite o adâncime de 2 nivele.

Setul de instrucţiuni pentru acest MC este dat în tabelul 5.2.

Tabelul 5.2

Setul de instrucţiuni PIC 12

Mnemonică-operanzi	Descriere
ADDWF f,d	Adună W cu f
ANDWF f,d	ŞI LOGIC între W şi f
CLRF f	Face în f toţi biţii 0
CLRW	Face în W toţi biţii 0
COMF f,d	Complementează f
DECF f,d	Decrementează f
DECFSZ f,d	Decrementează f şi trece mai departe dacă este 0
INCF f,d	Incrementează f
INCFSZ f,d	Incrementează f şi trece mai departe dacă este 0
IORWF f,d	SAU W cu f
MOVF f,d	Mută f
MOVWF f	Mută W în f
NOP	Nici o operaţie
RLF f,d	Rotaţie la stânga prin Carry
RRF f,d	Rotaţie la dreapta prin Carry
SUBWF f,d	Scădere W din f
SWAPF f,d	Schimbă f
XORWF f,d	SAU EXCLUSIV W cu f
BCF f,b	Bitul b din f este făcut 0
BSF f,b	Bitul b din f este făcut 1
BTFSC f,b	Se testează bitul b din f şi face salt dacă bitul este 0
BTFSS f,b	Se testează bitul b din f şi face salt dacă bitul este 1
ANDLW k	ŞI între W şi constanta k
CALL k	Chemare subrutină
CLRWDT k	Resetare ceas de gardă
GOTO k	Salt necondiţionat
IORLW k	SAU între W şi k
MOVLW k	Încărcare imediată a constantei k în W
OPTION k	Încărcare registru de opţiuni
RETLW k	Întoarcere din subrutină cu plasarea k în W
SLEEP	Intrarea în mod inactiv
TRIS f	Încărcarea registrului TRIS
XORLW k	SAU EXCLUSIV între k şi W

Dacă valoarea bitului d este 0, rezultatul se stochează în W iar dacă este 1, rezultatul se stochează în f.

Registrul de opțiuni (OPTION) este un registru special de configurare. Registrul TRIS este folosit pentru a controla liniile I/O. Un 1 în TRIS pune linia corespunzătoare în înaltă impedanță, iar un 0 validează linia.

Liniile de intrare/ieşire pot fi programate ca intrări sau ieşiri cu registrul special GPIO. Pinii pot avea semnificații duble. La RESET toate liniile se definesc ca intrări. Unii pini pot trezi MC din starea inactivă (Wake up). Operațiile de intrare ieşire se fac prin intermediul registrului GPIO, de exemplu instrucțiunea:

```
BCF     GPIO,5                    ;stabileşte un 0 pe linia 5 de ieşire
```

Modulul timer 0 poate fi utilizat în următoarele moduri:

- temporizator/numărător pe 8 biți;
- numărător pentru prescalare de 8 biți;
- ceas din exterior sau din interior.

Schema bloc a temporizatorului cu semnalele de comandă este dată în figura 5.2.

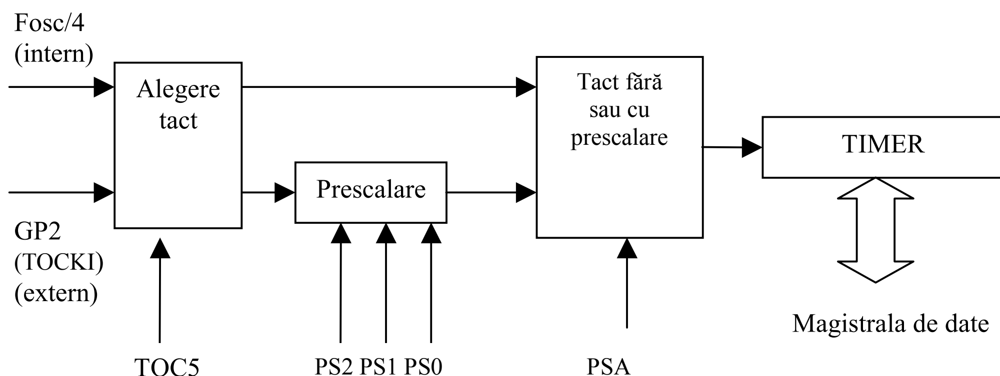


Figura 5.2 Schema bloc a temporizatorului PIC12

Cu bitul TOC5 se alege sursa tactului, externă (de la un pin cu semnificație dublă) sau internă. Tacul merge direct la timer sau prin registrul de prescalare (registru de 8 biți), programabil cu bitul PSA. Registrul de prescalare poate fi încărcat cu biții PS2, PS1 şi PS0. Toți aceşti biți de comandă se află în registrul de opțiuni. Registrul de prescalare poate fi folosit şi de ceasul de gardă, dar nu simultan cu timerul 0.

Memoria EEPROM cu care sunt echipate anumite circuite din familie poate fi de 16 octeți, poate fi supusă la peste 1 mil. de cicluri de scriere/ştergere şi poate reține informația mai mult de 40 de ani. Transmisia se face serial sincron pe două fire, unul de tact (SCL) şi unul de date, bidirecțional (SDA), mapate în registrul GPIO ca bit 6 şi bit 7, fără a avea conexiune în exterior. Protocolul pentru

transferul de date poate avea loc doar dacă magistrala de date nu este ocupată, cu SDA şi SCL=1. Un START este determinat de frontul SDA din 1 în 0, iar un front SDA din 0 în 1 reprezintă un STOP. Data trebuie să fie stabilă pe SDA pe durata unui impuls de tact SCL (o tranziție din 1 în 0, urmată de una din 0 în 1). Subrutinele de scriere/citire a EEPROM-ului sunt disponibile pe site-ul firmei.

Pentru a asigura siguranța rulării corecte a programului, circuitul este echipat cu un ceas de gardă, care poate fi resetat printr-o instrucțiune specială, asigurând o întârziere de 18ms până să declanşeze un RESET.

Familia PIC12 are posibilitatea de lucru într-un mod cu economie de energie, numit SLEEP. În modul SLEEP oscilatorul este oprit, iar pinii I/O îşi păstrează starea. Intrarea în SLEEP se face cu o instrucțiune specială. MC poate fi scos din acest mod de lucru printr-un front aplicat la pin extern sau de ceasul de gardă.

Ca şi mod de generare a tactului extern, circuitul poate lucra în mai multe feluri:

- cu cristal extern conectat la GP5/OSC1 şi GP4/OSC2 după o schemă standard;
- cu generator de tact extern, cuplat între GP5/OSC1 şi masă;
- pentru aplicații care nu sunt critice la timp, se poate conecta în exterior la GP5 un grup RC, cu R (3k-100k) la +5V şi C (20pF) la masă;
- MC are un generator intern de 4MHz, a cărui frecvență poate fi modificată prin scrierea registrului OSCCAL.

Alegerea sursei tactului, precum şi activarea ceasului de gardă se fac cu un registru de configurare (de 12 biți) care nu este accesibil utilizatorului, fiind o informație scrisă în PROM/ EPROM la adresa FFFh.

Un RESET poate fi generat de una din următoarele surse:

-la conectarea tensiunii de alimentare, POR Power On Reset;
-un RESET extern /MCLR pe pinul GP3;
-un RESET când circuitul este în mod SLEEP pe /MCLR;
-de la ceasul de timp real în operare normală;
-de la ceasul de timp real în mod SLEEP;
-trezirea din mod SLEEP prin schimbarea stării unui pin extern.

Identificarea sursei de RESET se face prin poziționarea unor biți în registrul de stare, registru al cărui conținut nu se modifică prin RESET.

Memoria poate fi programată în circuit (variantele cu EPROM). Aceasta se realizează simplu, cu 2 linii, una de date şi una de tact, conform figura 5.3.

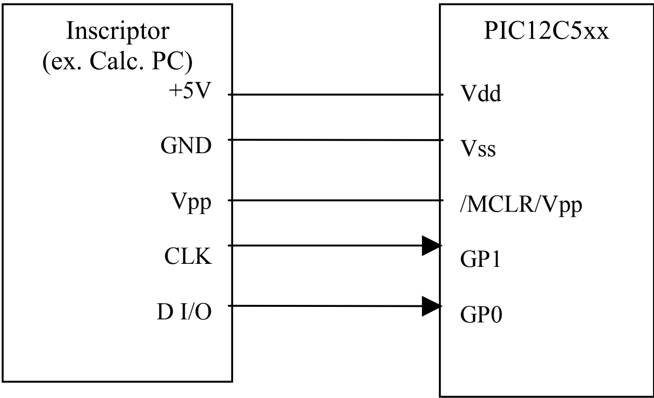


Figura 5.3 Conexiune pentru programarea EPROM

Intrarea în mod programare se face cu GP1 şi GP0 ţinuţi la 0 pe un front crescător al /MCLR.

Pentru a sublinia dimensiunea şi simplitatea de utilizare a acestui MC, în figura 5.4 este repreyentată capsula circuitului.

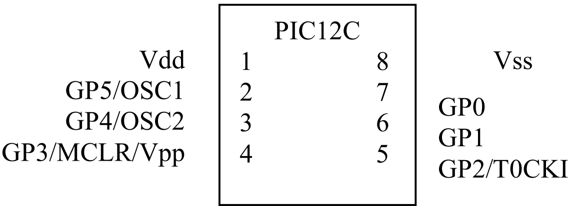


Figura 5.4 Capsula PIC12

5.1.2 PIC16

Cu o arhitectură asemănătoare familiei PIC12, aceste MC au câteva îmbunătăţiri:

- 35 de instrucţiuni faţă de 33;
- frecvenţa maximă 20MHz faţă de 4;
- posibilitatea de lucru în întreruperi, cu 7 surse interne şi o sursă externă;
- stivă automată cu 8 nivele;
- 13 linii I/O cu posibilitatea stabilirii individuale a sensului de transfer şi cu o linie de putere pentru comanda directă a unui LED;

- magistrala de instrucțiuni este pe 14 biți, față de 12 biți;
- memoria ROM este de 512 x 14 cuvinte până la 2K x 14 cuvinte;
- memoria RAM este de 80-128 de octeți.

Aceste MC sunt livrate în capsule cu 20 de pini.

Întreruperile sunt controlate de un registru de comandă a întreruperilor care poate valida sau invalida global sistemul de întreruperi şi individual pe fiecare linie. Fiecare interfață care poate cere întrerupere are ataşat un bit care poate fi testat pentru a determina sursa întreruperii.

Datorită posibilității de lucru în întreruperi, timerul 0, care funcționează la fel ca la PIC12 poate cere întrerupere la trecerea numărătorului de la FFh la 00h. Întreruperea poate fi mascată.

Pentru creşterea siguranței în funcționare familia PIC16 are integrat un circuit de protecție care generează un RESET la scăderea tensiunii de alimentare (Brown-Out Reset). Acest circuit poate fi validat sau invalidat cu un bit de comandă.

MC din familia PIC 16 pot fi echipate cu o diversitate mult mai mare de interfețe, cum ar fi comparatoare, convertoare ADC şi DAC, USART, I^2C, SPI, PWM etc. Circuitul PIC16C64x este echipat cu 2 comparatoare analogice. Intrările lor sunt multiplexate cu pinii I/O 0-3 şi cu o referință de tensiune care poate fi folosită pentru comaparatoare. Cu un registru de comandă în zona registrelor cu funcții speciale se poate programa modul de comparare. Se poate programa ca ieşirea comparatoarelor să ceară întrerupere. Comparatoarele pot fi programate să lucreze cu referință externă sau internă. În modul SLEEP comparatoarele rămân active şi pot trezi circuitul. Dacă curentul consumat de comparatoare în mod SLEEP este prea mare, ele pot fi dezactivate prin registrul de comandă. Modulul care generează referința de tensiune este un grup de 16 rezistențe care divizează tensiunea de intrare cu un factor programabil. Modulul poate fi validat sau invalidat pentru economia de energie.

Circuitul PIC16C71x este echipat cu convertor A/D şi un bloc suplimentar de timere. Convertorul A/D este un convertor cu aproximații succesive pe 8 biți, cu 4 intrări analogice multiplexate şi cu circuit de eşantionare-memorare. Tensiunea de referință poate fi cea de alimentare sau o referință externă la un pin I/O cu semnificație dublă. Convertorul A/D poate lucra şi în modul SLEEP, pentru acest mod de lucru fiind integrat în circuit un oscilator propriu pentru convertor. Convertorului îi sunt ataşate 3 registre, 2 de comandă şi unul de date. Cu registrele de comandă se poate programa:

- selecția tactului pentru convertor (Fosc/2, /8, /32 sau ceas propriu);
- selecția canalului analogic;
- un bit pentru START conversie;
- un bit pentru terminare conversiei - poate fi citit şi testat prin program, sau poate cere o întrerupere;
- un bit pentru oprirea convertorului pentru a nu mai consuma curent, dacă nu este folosit în mod SLEEP.

Modulul suplimentar de timere este numit CCP (Capture Compare PWM), după funcțiile pe care le poate îndeplini (un MC poate avea unul sau mai multe module CCP). Modulul CCP conține un registru de 16 biți şi foloseşte timerele suplimentare 1 şi 2. Modurile de lucru posibile pentru modulul CCP sunt:

- **mod captură -** la apariția unui eveniment la pinul exterior 3, registrul CCP se încarcă cu valoarea din timerul 1. Un eveniment poate fi un front crescător, unul descrescător, la fiecare 4 sau 16 fronturi crescătoare (prescalare). În momentul evenimentului se poate cere o întrerupere.
- **mod comparare** - registrul CCP este permanent comparat cu conținutul timerului 1. Dacă apare o coincidență, se semnalizează prin schimbarea stării pinului extern 3. În acelaşi moment se poate cere o întrerupere.
- **mod PWM** - se folosesc 2 timere, 1 şi 2, unul pentru a determina perioada semnalului şi celălalt factorul de umplere. Ieşirea PWM se face tot la pinul 3.

5.1.3 PIC17

PIC17 are următoarele îmbunătățiri față de PIC16:

- 58 de instrucțiuni față de 38;
- frecvența maximă 33MHz față de 20;
- stivă automată cu 16 nivele față de 8;
- 33 de linii I/O cu posibilitatea stabilirii individuale a sensului de transfer şi cu o linie de putere pentru comanda directă a unui LED;
- magistrala de instrucțiuni este pe 16 biți, față de 14 biți;
- memoria ROM este de 2048 x 16 cuvinte;
- memoria RAM este de 232 de octeți;
- magistralele de date, şi adrese (multiplexate) sunt accesibile la pin;
- unitatea centrală poate executa înmulțiri;
- sunt admise 11 surse de întreruperi față de 8.

Aceste MC sunt livrate în capsule cu 44 de pini.
Schema bloc este dată în figura 5.5.
Şi circuitele din familia PIC17 pot fi echipate cu o gamă largă de interfețe.

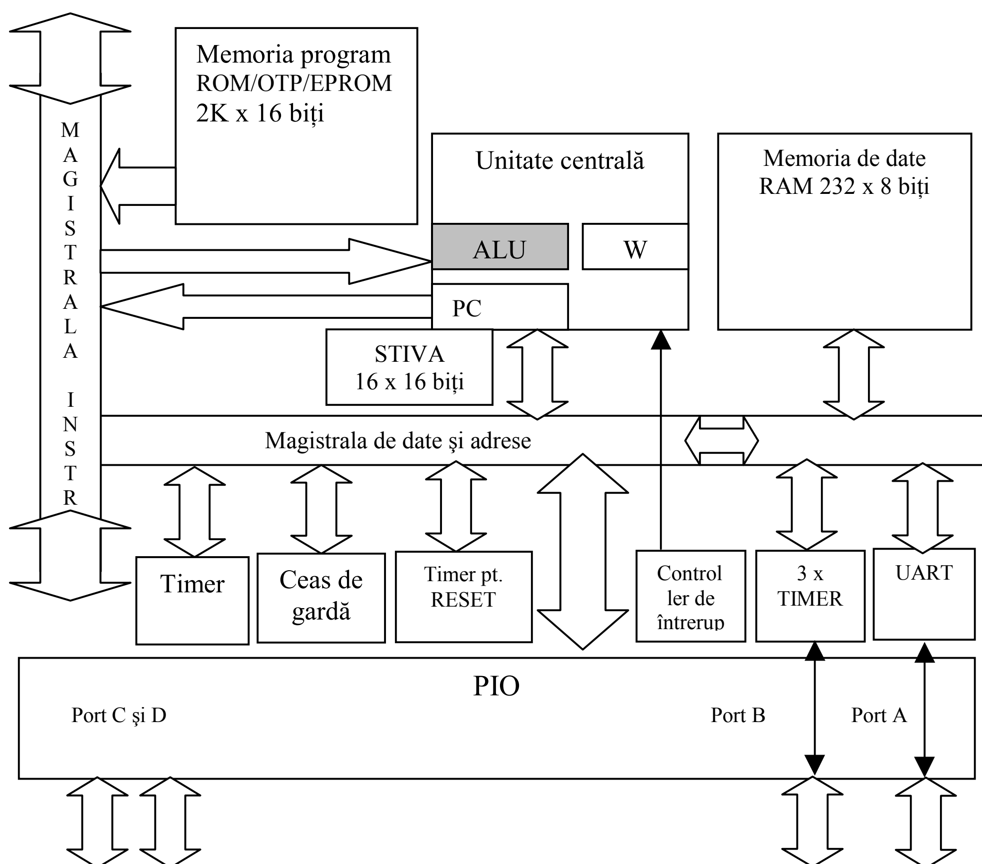


Figura 5.4 Schema bloc PIC17

Sistemul de timere este compus din mai multe timere:

- Timer 0 este un numărător de 16 biți cu sursă de numărare externă sau internă. Poate lucra cu un registru de prescalare care dă un tact divizat cu 1/1 până la 1/256;
- Timer 1 este un timer pe 8 biți care poate lucra împreună cu Timerul 2 pentru a forma un timer pe 16 biți. Poate lucra în regim de numărare a unor evenimente externe. Timerul are asociat un registru, deci poate lucra în regim de comparare;
- Timer 2 este la fel cu timerul 1. Timerul 1 şi 2 pot lucra în mod PWM;
- Timer 3 este pe 16 biți, cu un registru asociat.

Modulul USART poate fi configurat să lucreze în mod serial asincron full-duplex (pentru a transfera date cu un terminal sau un PC), sau în mod sincron full-duplex (de ex. pentru a lucra cu memorii externe EEPROM).

Observație: este de remarcat la MC din familia PIC faptul că nu au nevoie de componente externe pentru RESET şi pentru ceas, ceea ce poate fi un avantaj.

5.2 MC ATMEL

Foarte bine plasate pe piață în zona aplicațiilor low cost sunt MC ale firmei ATMEL. Firma fabrică şi MC echivalente 8051.

5.2.1 Familia AVR

Aceste MC sunt realizate de producător în diverse variante, începând cu modelele low cost AT90S2323, AT90S1200 şi terminând cu modele complexe AT90S4414, AT90S8515, diferențele între acestea constând în principal în mărimea şi tipurile memoriei RAM, FLASH, ROM, EPROM, EEPROM şi a facilităților oferite la interfața cu utilizatorul.

Seria de MC ATMEL este susținută de o campanie puternică de promovare astfel încât producătorul oferă gratuit software de dezvoltare ce include asamblor, debugger, documentație, exemple şi note de aplicație pe site-ul său www.atmel.com. Softul este disponibil atât în variantă ce rulează sub sistemul de operare MS-DOS cât şi în varianta WINDOWS.

Toate MC din familie au o arhitectură Harvard, adică au spații de adresă şi magistrale diferite pentru memoria de date şi memoria program. Dispun de o prelucrare de tip pipe line a instrucțiunilor (în 2 trepte) astfel încât în timp ce o instrucțiune este executată, cealaltă se află în ciclu de aducere din memorie (fetch) şi astfele se execută câte o instrucțiune în fiecare ciclu.

Schema bloc a unui circuit din familie este dată în figura 5.5.

Unul dintre cele mai mici MC este AT90S2323. Circuitul se alimentează (în funcție de variantă) la 5V sau la 3V şi funcționează până la frecvența de 10MHz. Curentul absorbit este de 2,4mA în stare activă, 0,5mA în stare inactivă şi 1µA în starea Power Down. În principal, circuitul este compus din:

- unitatea centrală, în arhitectură RISC cu 118 instrucțiuni, majoritatea de un ciclu şi 32 de registre de uz general;
- blocul de memorie, compus din memoria de program (FLASH de 2K octeți care suportă în jur de 1000 de programări) şi din memoria de date (128 octeți de RAM) şi 128 octeți de EEPROM, care suportă în jur de 100.000 de programări;

- interfețele sunt reprezentate de un timer de 8 biți cu prescalare, un timer pentru ceasul de gardă şi o interfață serială SPI pentru programarea în circuit. Liniile I/O sunt de regulă cu semnificație dublă.

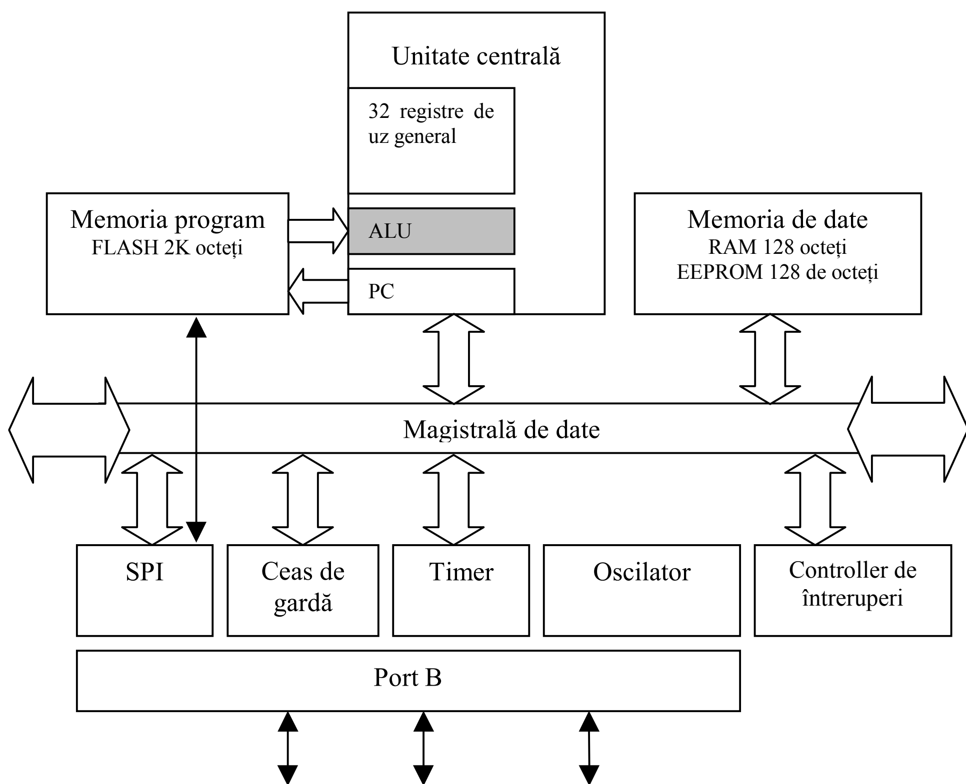


Figura 5.5 MC din familia AVR – schema bloc

Unitatea centrală poate executa majoritatea operațiilor într-un ciclu, ceea ce înseamnă că operanzii sunt în registrele generale, are loc operația şi rezultatul este stocat în unul dintre registre. Se pot realiza şi adresări indirecte cu 6 dintre cele 32 de registre, grupate câte 2 (ca să formeze registre de 16 biți); registrele duble sunt referite cu X, Y, Z. Se pot face operații între registre sau cu o constantă. Instrucțiunile au formatul pe 16 biți, iar indicatorul de program PC este pe 10 biți.

Memoria de date poate fi accesată tot ca registre, în acelaşi spațiu. Spațiul I/O conține 64 de adrese unde se găsesc registrele de control şi de stare ale interfețelor. Stiva este definită în RAM, deci există un registru indicator de stivă SP de 8 biți. O imagine sugestivă a spațiilor de memorie şi I/O este dată în figura 5.6.

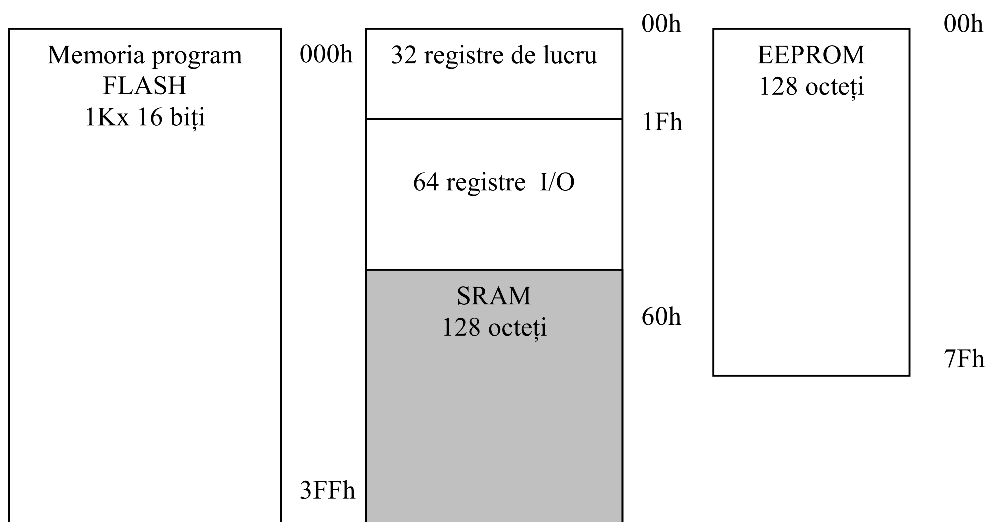


Figura 5.6 Harta spațiilor de memorie şi I/O

Memoria EEPROM dispune de un spațiu propriu de adresare, în care fiecare octet dintr-o locație poate fi citit sau scris. Accesul se face specificând adresa, data şi comanda în registre speciale.

Acest MC permite 5 moduri de adresare:

- adresare directă - adresa operandului este conținută în instrucțiune. Se pot executa instrucțiuni cu un registru sau între 2 registre. Este accesibil astfel tot spațiul de date;
- adresare indirectă - adresa operandului este în X, Y sau Z;
- adresare indirectă cu deplasament - adresa oprandului este rezultatul adunării registrelor Y sau Z cu adresa de 6 biți conținută în instrucțiune. Se acoperă astfel doar 63 de locații față de baza dată de registrele Y sau Z;
- adresare indirectă cu pre decrementare - registrele X, Y sau Z sunt decrementate înainte de adresare;
- adresare indirectă cu post incrementare.

Setul de instrucțiuni este dat în tabelul 5.3.

Tabelul 5.3

Setul de instrucțiuni al familiei AVR

Mnemonică	Operanzi	Operație	Operație
ADD	Rd,Rr	Rd=Rd+Rr	Sumă
ADC	Rd,Rr	Rd=Rd+Rr+C	Sumă plus carry
SUB	Rd,Rr	Rd=Rd-Rr	Diferență
SUBI	Rd,K	Rd=Rd-K	Diferență cu o constatntă
SBC	Rd,Rr	Rd=Rd-Rr-C	Diferență cu carry
SBCI	Rd,K	Rd=Rd-K-C	Dif. cu const. şi carry
AND	Rd,Rr	Rd=Rd*Rr	SI logic
ANDI	Rd,K	Rd=Rd*K	SI logic cu const.
OR	Rd,Rr	Rd=Rd v Rr	SAU logic
ORI	Rd,K	Rd=Rd v K	SAU logic cu const.
EOR	Rd,Rr	Rd=Rd⊕Rr	SAU exclusiv
COM	Rd	Rd=$FF-Rd	Complement față de 1
NEG	Rd	Rd=$00-Rd	Complement față de 2
SBR	Rd,K	Rd=Rd v K	Set bit K în registru
CBR	Rd,K	Rd=Rd*(FFh-K)	Clear bit K în registru
INC	Rd	RD=Rd+1	Increment
DEC	Rd	Rd=Rd-1	Decrement
TST	Rd	Rd=Rd * Rd	Test de 0 sau -
CLR	Rd	Rd=Rd⊕Rd	Clear registru
SER	Rd	Rd=$FF	Set registru
RJMP	k	PC=PC+k+1	Salt relativ la k
RCALL	k	PC=PC+k+1	Salt la subrutină
RET		PC=STACK	Return din subrutină
RETI		PC=STACK	Ret. din subrutină de tratare a întreruperii
CPSE	Rd,Rr	if (RD=Rr) PC=PC+2 or 3	Compară , skip dacă egal
CP	Rd,Rr	Rd-Rr	Compară
CPC	Rd,Rr	Rd-Rr-C	Compară cu carry
CPI	Rd,K	Rd-K	Compară cu constată
SBRC	Rr,b	if (Rr(b)=0) PC=Pc+2 or 3	Skip dacă bitul b din Rr este 0
SBRS	Rr,b	if (Rr(b)=1) PC=Pc+2 or 3	Skip dacă bitul b din Rr este 1
SBIC	P,b	if (P(b)=0) PC=Pc+2 or 3	Skip dacă bitul B din I/O este 0
SBIS	P,b	if (P(b)=1) PC=Pc+2 or 3	Skip dacă bitul B din I/O este 1

Mnemonică	Operanzi	Operație	Operație
BRBS	s,k	if(SREG(s)=1) PC=PC+k+1	Salt dacă SF este1
BRBC	s,k	if(SREG(s)=0) PC=PC+k+1	Salt dacă SF este 0
BREQ	k	if (Z=1) PC=PC+k+1	Salt: egalitate
BRNE	k	if (Z=0) PC=PC+k+1	Salt: diferit
BRCS	k	if (C=1) PC=PC+k+1	Salt: carry setat
BRCC	k	if (C=0) PC=PC+k+1	Salt: carry este zero
BRSH	k	if (C=0) PC=PC+k+1	Salt: mai mare sau egal
BRLO	k	if (C=1) PC=PC+k+1	Salt: mai mic
BRMI	k	if (N=1) PC=PC+k+1	Salt: minus
BRPL	k	if (N=0) PC=PC+k+1	Salt: plus
BRGE	k	if (N⊕V=0) PC=PC+k+1	Salt : mai mare sau egal, cu semn
BRLT	k	if (N⊕V=1) PC=PC+k+1	Salt : mai mic decât 0, cu semn
BRHS	k	if (H=1) PC=PC+k+1	Salt dacă CF este setat
BRHC	k	if (H=0) PC=PC+k+1	Salt dacă CF este zero
BRTS	k	if (T=1) PC=PC+k+1	Salt dacă T este setat
BRTC	k	if (T=0) PC=PC+k+1	Salt dacă T este zero
BRVS	k	if (V=1) PC=PC+k+1	Salt dacă este overflow
BRVC	k	if (V=0) PC=PC+k+1	Salt dacă nu este overflow
BRIE	k	if (I=1) PC=PC+k+1	Salt dacă întreruperea e activată
BRID	k	if (I=0) PC=PC+k+1	Salt dacă întreruperea e dezactivată
LD	Rd,Z	Rd=(Z)	Incarcă registru indirect
ST	Z,Rr	(Z)=Rr	Stochează registru indirect
MOV	Rd,Rr	Rd=Rr	Mută Rr în Rd
LDI	Rd,K	Rd=K	Mută constanta K în Rd
IN	Rd,P	Rd=P	IN din portul P
OUT	P,Rr	P=Rr	OUT la portul P
SBI	P,b	I/O (p,B)=1	Set bit b din portul P
CBI	P,b	I/O (p,B)=0	Reset bit b din portul P
LSL	Rd	Rd(n+1)=Rd(n),Rd(0)=0	Shift logic stânga
LSR	Rd	Rd(n)=Rd(n+1),Rd(7)=0	Shift logic dreapta
ROL	Rd	Rd(0)=C,Rd(n+1)=Rd(n), C=Rd(7)	Rotire spre stânga prin carry

Mnemonică	Operanzi	Operație	Operație
ROR	Rd	Rd(7)=C,Rd(n)=Rd(n+1), C=Rd(0)	Rotire spre dreapta prin carry
ASR	Rd	Rd(n)=Rd(n+1), n=0..6	Shift aritmetic spre dreapta
SWAP	Rd	Rd(3..0)=Rd(7..4), Rd(7..4)=Rd(3..0)	Inversează jumătățile
BSET	s	SREG(s)=1	Setează flag s
BCLR	s	SREG(s)=0	Resetează flagul s
BST	Rr,b	T=Rr(b)	Stochează bitul b din Rr în T
BLD	Rd,b	Rd(b)=T	Incarcă T în bitul b din Rd
SEC		C=1	Setează Carry flag
CLC		C=0	Reset Carry flag
SEN		N=1	Setează Negative flag
CLN		N=0	Resetează Negative flag
SEZ		Z=1	Setează Zero flag
CLZ		Z=0	Resetează Zero flag
SEI		I=1	Activează întrerupere
CLI		I=0	Dezactivează întrerupere
SES		S=1	Setează flagul de semn
CLS		S=0	Resetează flagul de semn
SEV		V=1	Setează flagul overflow complement față de 2
CLV		V=0	Resetează flagul overflow complement față de 2
SET		T=1	Setează T în SREG
CLT		T=0	Resetează T în SREG
SEH		H=1	Setează Half Carry Flag în SREG
CLH		H=0	Resetează Half Carry Flag în SREG
NOP			No operation
SLEEP			Sleep
WDR			Watch dog reset

Anumite modele au şi o instrucțiune de înmulțire, MUL.
MC AVR au 3 surse de întrerupere:

- de la RESET, (din exterior, la punerea sub tensiune sau de la ceasul de gardă);
- de la un pin extern;
- de la timer, la o depăşire.

Întreruperile pot fi mascate cu 2 registre de 8 biți, GIMSK- General Interrupt Mask şi TIMSK- Timer/Counter Interrupt Mask. La primirea unei întreruperi se invalidează sistemul de înteruperi. Totuşi este posibilă primirea încă a unei întreruperi în timp ce este servită prima, dacă se revalidează sistemul de întreruperi. Răspunsul la o cerere de întrerupere durează minimum 4 cicluri, timp în care PC este salvat în stivă, SP este incrementat cu 2 şi se setează invalidarea întreruperilor.

Sursele de RESET pot fi:

- la punerea sub tensiune (Power On Reset), dacă tensiunea creşte şi atinge un anumit prag;
- de la un pin extern, dacă un nivel LOW este prezent mai mult de 50ns;
- de la ceasul de gardă dacă este validat şi a expirat perioada de timp pentru care a fost programat să aştepte o inițializare.

După RESET execuția programului începe de la adresa 000h. Un RESET porneşte un numărător care contorizează un anumit număr de impulsuri ale ceasului intern al circuitului. Contorul stabileşte durata impulsului RESET intern pentru ca acesta să fie suficient de lung pentru inițializarea tuturor circuitelor interne, figura 5.7.

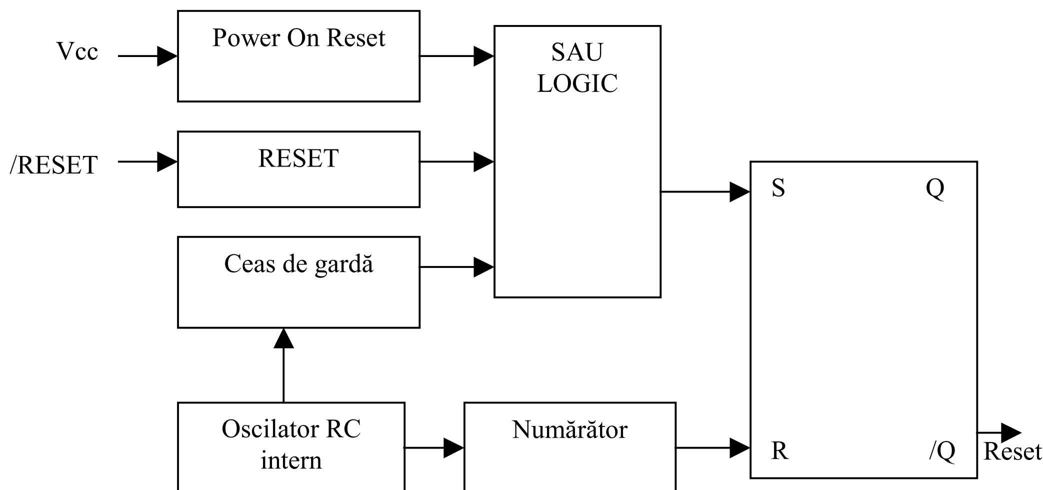


Figura 5.7 Circuit intern pentru generarea semnalului RESET

Nu mai este astfel necesară nici o componentă exterioară pentru semnalul de RESET.

MC admite moduri de lucru cu economie de energie:

- modul adormit (Power Down), în care se intră prin execuția instrucțiunii SLEEP. Dacă în modul SLEEP apare o întrerupere externă, una de la ceasul de gardă sau un RESET, acestea sunt executate, circuitul trezindu-se. În acest mod de lucru este oprit

oscilatorul extern. Trezirea durează un interval de timp egal cu cel necesar pentru un RESET.

- modul inactiv, în care se intră cu aceeaşi instrucţiune, dar poziţionând un bit din registrul de control. În acest mod unitatea centrală este oprită, dar timerul, ceasul de gardă şi sistemul de întreruperi continuă să funcţioneze. O întrerupere internă, una externă sau un RESET trezesc circuitul.

Circuitul admite ca surse de tact:

- un cristal de cuarţ sau un rezonator ceramic, pentru aplicaţii cu pretenţii la stabilitatea frecvenţei, (elemente conectate la pinii Xtal1 şi Xtal2);
- un generator extern la un pin I/O (PB3);
- generatorul RC intern, la 1MHz.

Selecţia sursei ceasului pentru tact intern sau extern se face cu un bit în memoria FLASH. Circuitul este programat implicit pentru tact extern. Lucrul cu ceas intern determină posibilitatea utilizării acestor MC cu componente externe extrem de puţine.

Timerul care echipează circuitele AT902323 este un timer pe 8 biţi cu un registru de prescalare pe 10 biţi. Timerul poate fi folosit cu tact intern (tactul sistemului divizat cu 8, 64, 256 sau 1024) sau cu tact extern. Sursa de tact pentru timer poate fi ceasul sistemului, ceasul prescalat sau un tact extern. Funcţionarea timerului este controlată de biţi din din două registre (registru de măşti pentru întreruperi şi registru de control al timerului). La depăşire, numărătorul poate cere o întrerupere.

Ceasul de gardă are ca şi tact un oscilator separat, integrat în MC şi este complet separat de timer. Ca şi timerul, ceasul de gardă are posibilitatea de prescalare. Controlul ceasului de gardă se face cu registrul WDTCR (Watchdog Timer Control Register). Pentru a evita dezactivarea întâmplătoare a ceasului de gardă, acesta trebuie dezactivat cu o secvenţă de program specifică.

EEPROM este accesibil în spaţiul I/O, prin următoarele registre:

- registru de adrese EEPROM;
- registru de date EEPROM;
- registru de control EEPROM, care comandă sensul transferului.

MC ATMEL AVR au linii I/O de uz general (AT902323 are 3 linii, AT902343 are 5 linii, iar modelul AT90S1200 dispune de două porturi: B de 8 biţi şi D de 7 biţi). Fiecare linie de port poate fi configurată independent faţă de celelalte, atât ca linie de intrare cât şi ca linie de ieşire. De asemenea în modul de utilizare ca linie de intrare, fiecare linie poate fi configurată cu un rezistor de pull-up intern (valoarea aprox 40KΩ). Fiecare linie de ieşire poate susţine un curent de 20mA (max 40mA valoare limită absolută) astfel încât se poate folosi direct la

comandarea de LED-uri. Fiecărui port îi este alocat un registru de sens, (DDRx, Port x Data Direction). Pinii I/O admit semnificații duble.

Programarea memoriei FLASH şi EEPROM se poate face serial, într-un mod cu tensiune mare (12V) şi un mod cu tensiune joasă. Circuitele vin de la fabricant gata de a fi înscrise (au conținutul FFh în fiecare locație). În modul de programare cu tensiune înaltă, tensiunea de 12V validează programarea, nu are un rol funcțional. Modul de programare în cele 2 variante este arătat în figura 5.8.

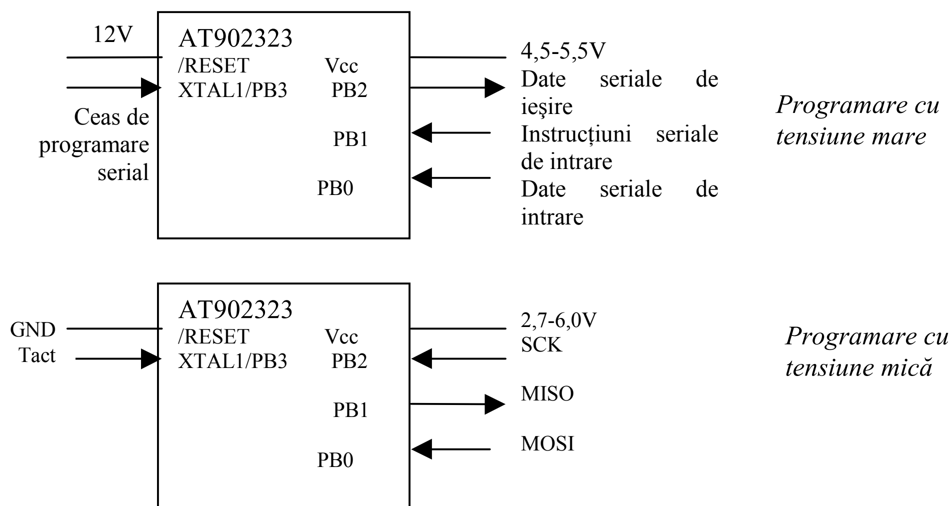


Figura 5.8 Programarea serială a memoriei EPROM (FLASH)

La programarea cu tensiune mare se poate programa memoria FLASH (intrarea pe PB1) şi memoria de date EEPROM (intrarea pe PB0). Pentru programarea memoriei FLASH se trimite întâi adresa apoi data (octetul LOW apoi cel HIGH). Confirmarea se obține prin trecerea lui PB2 în HIGH. Memoria EEPROM se programează trimițând întâi adresa apoi octetul de date, confirmarea fiind pe pinul PB2. Orice locație poate fi citită folosind instrucțiunea de citire şi adresa, obținând pe pinul PB2 conținutul respectiv. Tactul serial este activ pe front crescător. Scrierea, citirea şi unele comenzi speciale (ştergerea întregii memorii, scrierea biților de securitate etc.) se comandă prin trimiterea înaintea adresei a codului serial al comenzii respective (se găseşte în foile de catalog).

La programarea cu tensiune mică se poate programa memoria FLASH şi memoria EEPROM tot serial, prin interfața SPI. Datele se înscriu pe frontul crescător a lui SCK. În acest mod de lucru, confirmarea scrierii unei locații se face prin trimiterea în ecou a octetului scris. Comenzile se trimit la fel, prin coduri seriale.

Un programator pentru astfel de MC (mod de programare cu tensiune mare) se poate realiza simplu, folosind interfața CENTRONICS a unui PC, figura 5.9.

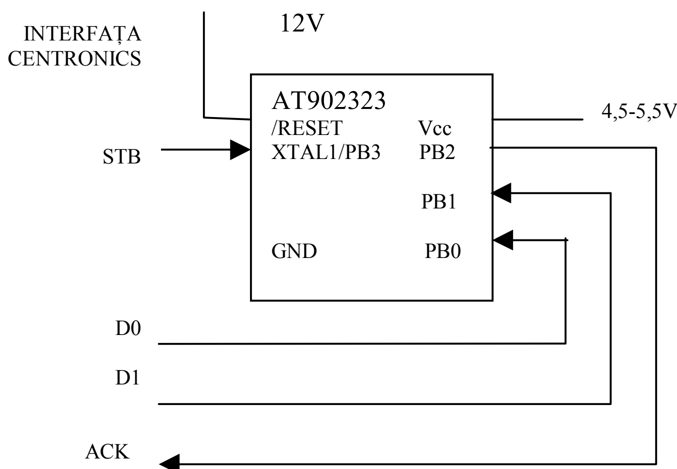


Figura 5.9 Programarea unui circuit AVR folosind interfața CENTRONICS

Memoria este prevăzută cu posibilitatea de protecție a programelor înscrise. Astfel, există 2 biți de blocare în FLASH care nu pot fi şterşi decât prin ştergerea întregului program. Aceşti doi biți pot comanda:

- dezactivarea unor viitoare programări ale memoriei FLASH sau EEPROM;
- dezactivarea unor viitoare programări ale memoriei FLASH sau EEPROM şi blocarea verificării.

Memoria mai este prevăzută cu 2 biți care nu pot fi şterşi (fuzibili) care pot comanda:

- dezactivarea modului serial de programare;
- sursa internă sau externă pentru tact.

Modelul ATiny 10 este echipat cu un comparator analogic, care compară valoarea analogică de la pinul PB0 cu cea de la pinul PB1. Ieşirea comparatorului poate declanşa o întrerupere. Comparatorul este controlat de un registru de control şi stare.

Modelul AT90S4433 este echipat cu 6 canale de conversie A/D pe 10 biți. Acest model este un MC cu o echipare superioară: un timer suplimentar de 16 biți cu posibilitatea de comparare, captură şi generare PWM, un comparator analogic, un canal UART, un convertor A/D pe 10 biți cu 6 canale şi un sistem de întreruperi care admite 14 surse de întrerupere din care 2 externe. Acest circuit are şi posibiliataea de programare paralelă, deoarece are suficiente linii I/O. Schema bloc a acestui MC este dată în figura 5.10.

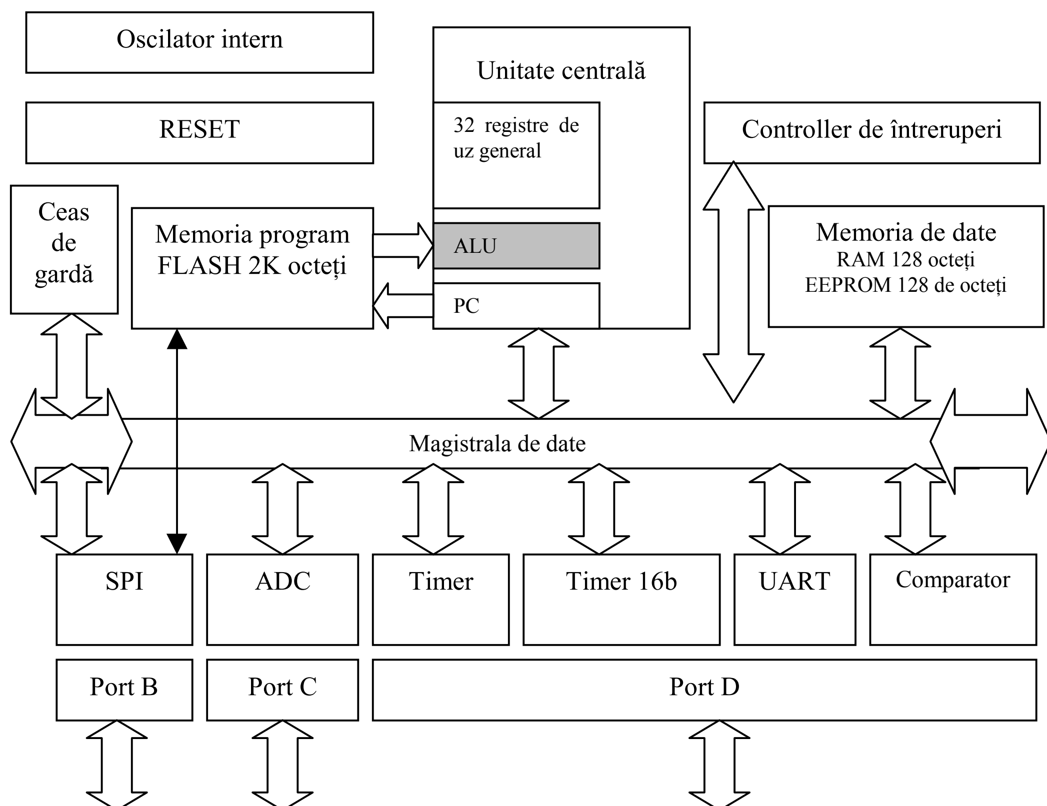


Figura 5.10 MC AT90S4433 – Schema bloc

Canalul serial UART permite transferul de date full duplex, cu 8 sau 9 biți de date, cu generarea de rată de transfer. Canalul serial poate cere întrerupere la transmisie completă, registru de transmisie gol sau recepție completă. Recepția şi transmisia se fac ca şi la celelalte MC, cu un registru de deplasare pentru serializare/deserializare şi un registru de date. Se poate realiza şi o comunicație serială multiprocesor (Multi- Processor Communication Mode), prin transmisia în mesaj întâi a adresei unui destinatar şi apoi a mesajului propriu-zis. Se poate astfel realiza o comunicație în care un procesor este MASTER iar celelalte SLAVE. Canalul serial este controlat de un registru de comandă şi stare.

Comparatorul analogic, pe lângă întreruperea pe care o poate genera, poate să declanşeze o captură la timerul de 16 biți la acest tip de MC.

Convertorul A/D este un convertor cu aproximații succesive cu eşantionare memorare şi cu un bloc de multiplexare analogică pe 6 canale la intrare. Convertorul poate lucra cu conversie singulară sau conversie continuă. Convertorul are pini de alimentare separați din exterior şi un pin pentru conectarea tensiunii de referință. Ca şi tact de conversie, convertorul acceptă tactul sistemului divizat cu un registru de prescalare. Tactul se poate încadra în valorile optime 50-200kHz.

La terminarea unei conversii convertorul poate cere o întrerupere. Schema bloc a convertorului este dată în figura 5.11.

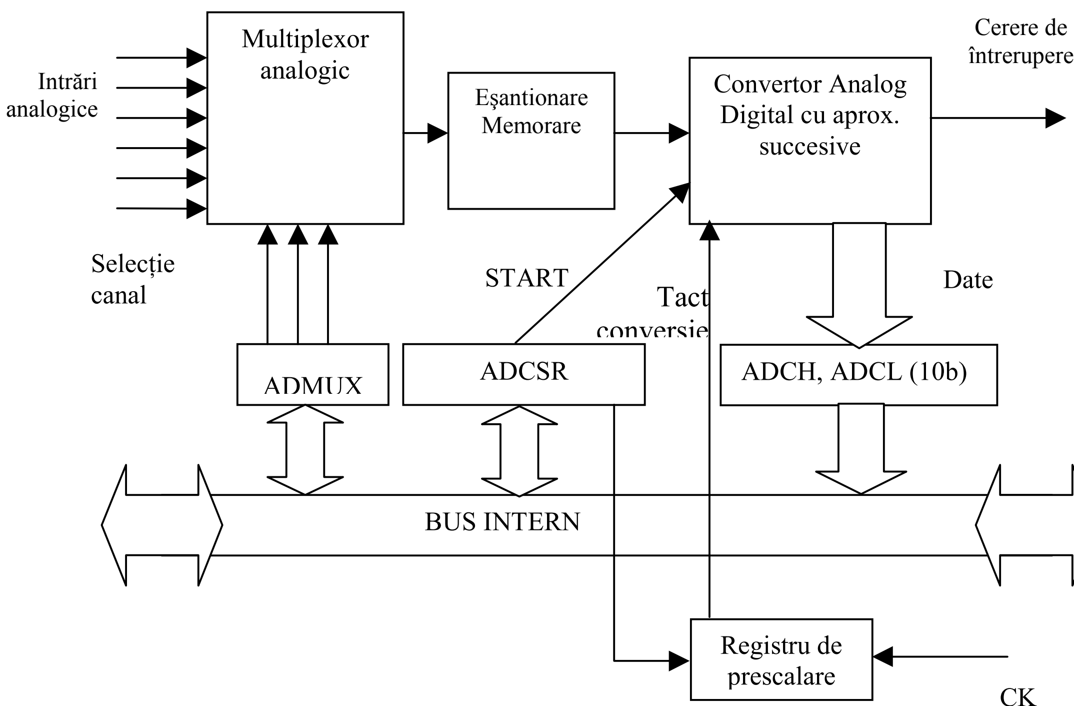


Figura 5.11 Convertorul analog-numeric

Prin registrul ADMUX se comandă cu 3 biți selecția unui canal din cele 6. Registrul de control şi stare ADCSR poate declanşa o conversie prin poziționarea unui bit. Funcționarea convertorului poate fi validată/invalidată. Prin registrul de control se poate programa şi rata de prescalare. Pentru a micşora perturbațiile introduse de unitatea centrală în timpul achiziției de date se poate comanda UC în stare inactivă. După efectuarea conversiei întreruperea de la convertor va trezi UC (ADC Noise Canceler Function).

Înscrierea memoriei FLASH şi EEPROM se poate face şi în mod paralel, ceea ce înseamnă o viteză mai mare de scriere şi un plus de simplitate la programare. Semnalele folosite în acest mod de programare sunt date în figura 5.12.

Pentru înscrierea sau citirea datelor se folosesc 8 linii cu semnificații duble din porturile PC şi PB. /OE stabileşte dacă este vorba de scriere sau citire. Scrierea se face cu strobul /WR. BS selectează octetul mai semnificativ sau mai puțin semnificativ. Semnalele XA0 şi XA1 stabilesc dacă se încarcă memoria FLASH, sau o adresă a EEPROM, un octet de date sau o comandă. Sunt posibile 9 comenzi:

ştergere memorie, scriere biţi de securitate, scriere biţi fuzibili, scriere FLASH, scriere EEPROM, citire biţi de identificare, citire biţi fuzibili şi de securitate, citire FLASH, citire EEPROM. Modul de scriere paralel poate fi uşor realizat prin intermediul interfeţei CENTRONICS; citirea pentru verificare este mai complicată.

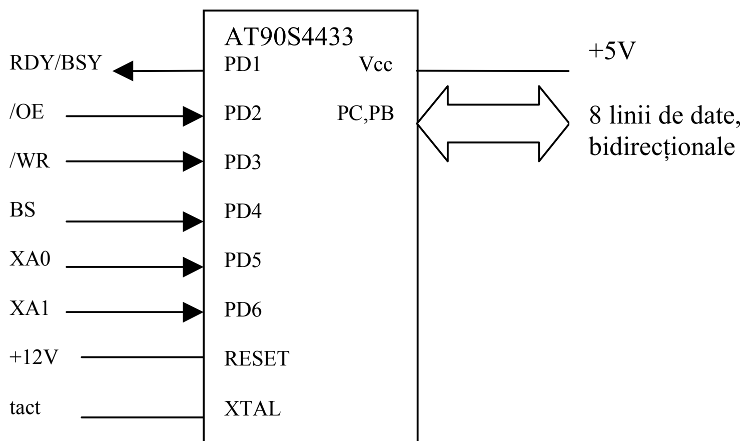


Figura 5.12 Semnale folosite la înscrierea paralelă a memoriei EEPROM (FLASH)

Modelele AT90S4414 şi AT90S8515 dispun de posibilitatea conectării în exterior a unei memorii SRAM suplimentare. Pentru aceasta sunt disponibile la portul A magistrala de adrese (octet mai puţin semnificativ) şi magistrala de date multiplexate, iar la portul C octetul mai semnificativ al magistralei de adrese. De asemenea sunt disponibile şi semnalele ALE (Address Latch Enable), /RD şi /WR.

5.2.2 Familia ARM

Un nucleu cu arhitectură RISC de mare performanţă este ARM7DMI. Este conceput cu o arhitectură von Neumann, cu magistrala de date pe 32 de biţi, având 2 seturi de instrucţiuni (ARM pe 32 de biţi şi THUMB pe 16 biţi). Spaţiul adresabil este de 4G.

O schemă bloc sumară a acestui nucleu este dată în figura 5.13.

Structura UC este pipe line pe 3 nivele; extragere cod, decodificare şi execuţie. UC poate lucra cu date pe 8, 16 sau 32 de biţi şi poate executa înmulţiri într-un singur ciclu. Nucleul are integrat şi circuitul de testare JTAG şi un emulator în circuit.

Setul de instrucţiuni THUMB pe 16 biţi este un subset al setului pe 32 de biţi, cele mai uzual folosite instrucţiuni. În acest fel se salvează spaţiu de memorare şi se câştigă viteză de prelucrare.

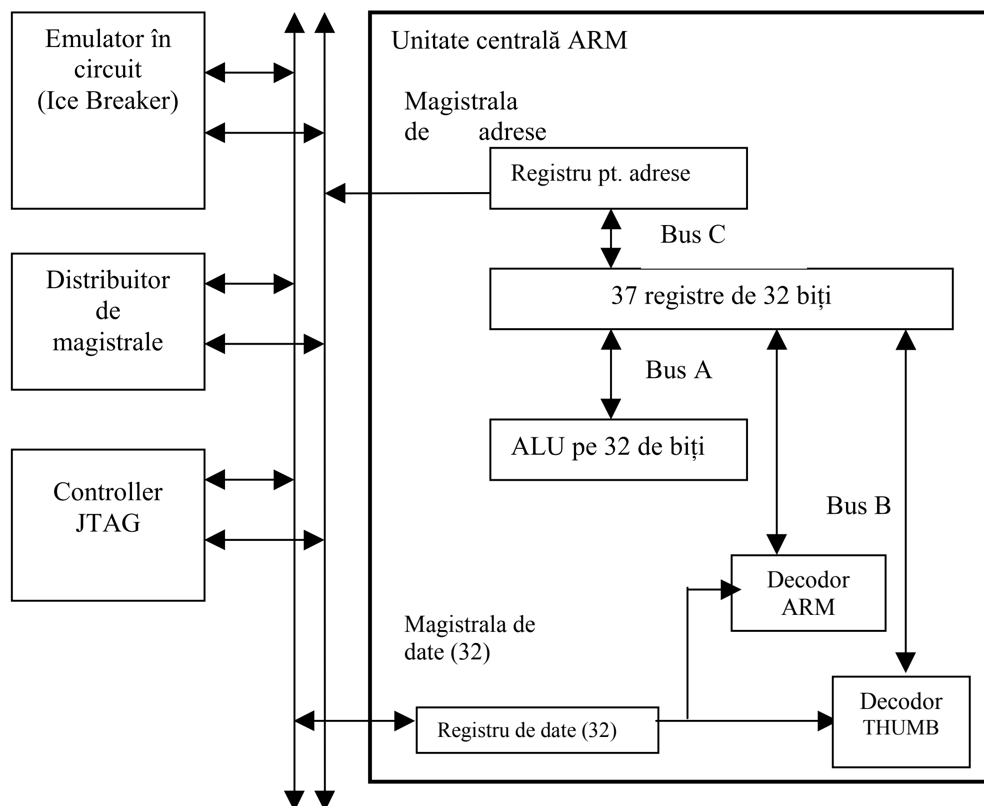


Figura 5.13 MC ARM – schema bloc

Pentru a asigura un grad mare de paralelism în UC, există mai multe magistrale de legătură. Instrucțiunile sunt analizate şi distribuite decodoarelor corespunzătoare (THUMB sau ARM). UC este echipată cu un set de 32 de registre şi registre suplimentare pentru înmulțire. Pentru a se putea conecta o varietate cât mai mare de interfețe, magistralele sunt accesibile prin distribuitor în mai multe forme: unidirecțional, bidirecțional, multiplexat etc.

6

CRITERII DE PROIECTARE

6.1 CRITERIILE PENTRU ALEGEREA UNUI MC

Sunt multe aspecte de care trebuie ținut seama la alegerea unui MC pentru o anumită aplicație. Alegerea unui MC potrivit poate duce la succesul proiectului, aşa cum o alegere nepotrivită poate duce la eşecul proiectului. Fiecare cititor trebuie să adapteze aceste criterii nevoilor sale şi scalei proprii de valori.

Obiectivul urmărit în alegerea unui MC este obținerea calității dorite cu un cost cât mai scăzut. Calitățile dorite înseamnă performanță, fiabilitate, calități EMC (de compatibilitate electromagnetică cu mediul), iar costul total include costurile cercetării, proiectării, construcției, testării, reparării produsului.

În primul rând se pune problema stabilirii funcției pe care MC trebuie s-o îndeplinească în sistem. Alegerea din catalog a unui MC trebuie făcută în ideea a cât mai puțin hardware suplimentar (din motive economice). Procesul de căutare este dificil din cauza numărului foarte mare de tipuri de MC disponibile pe piață. Munca de căutare este ajutată de bazele de date din Internet, aşa cum este baza de date de la www.questlink.com . După stabilirea MC optim se verifică prețurile, dacă este disponibil, suportul acordat de fabricant, existența uneltelor de dezvoltare, stabilitatea firmei constructoare. Un criteriu important este posibilitatea de a fi găsit pe piață (optenabilitatea), mai ales în zone în care circulația mărfurilor este destul de greoaie.

Criteriile pentru alegerea unui MC sunt, în ordinea importanței:

1.Posibilitatea folosirii în aplicația dată

- este suficient un MC sau sunt necesare circuite suplimentare;
- liniile I/O sunt suficiente (un număr prea mic înseamnă că aplicația nu se poate face cu acest MC, iar un număr prea mare înseamnă un cost excesiv);
- există toate interfețele solicitate de aplicație: I/O serial, convertoare A/D, D/A şi nu există interfețe în plus;
- există capacitatea de memorare suficientă: RAM, ROM;
- MC are viteza suficientă pentru această aplicație. Se verifică timpul necesar rulării programului care trebuie să fie mai mic decât intervalul de timp în care trebuie să reacționeze MC;

- alimentarea MC poate fi făcută din aplicaţie (este posibil ca aplicaţia să fie portabilă, atunci este nevoie de un MC care să funcţioneze la 3V;
- preţul acestui MC este bun (acceptabil) pentru aplicaţia respectivă.

2. Optenabilitatea MC

- trebuie să fie disponibil în cantităţi suficiente;
- trebuie să fie în producţia actuală, dar şi în viitor pentru posibilitatea aprovizionării în viitor;
- disponibilitatea unor accesorii (convertoare A/D, D/A, alimentatoare etc).

3.Disponibilitatea suportului de dezvoltare

- asambloare;
- compilatoare;
- debuggere;
- module de evaluare;
- emulatoare în circuit;
- analizoare logice;

4. Suport din partea constructorului

- documentaţie tehnică;
- buletine de aplicaţii;
- service prin telefon (BBS);
- rapoarte despre probleme de funcţionare;
- software de utilizare;
- dacă MC este folosit şi de alţi utilizatori, atunci sunt formate grupuri de lucru care pot oferi ajutor.

5.Seriozitatea constructorului

- dacă este demonstrată competenţa lui ;
- stabilitate şi fiabilitatea MC realizate;
- viteza de livrare;
- număr de ani ca şi constructor şi rezultate financiare.

Un argument pentru alegerea unui tip de MC este existenţa unui modul de evaluare. Pentru a promova propriile MC, mulţi furnizori au creat Kit-uri de evaluare care conţin plăci de evaluare şi un soft minimal cu care se poate învăţa utilizarea MC şi se pot pune la punct aplicaţii. Un kit conţine de regulă un program monitor pentru calculator PC, un program de transfer al datelor spre placa de evaluare (prin interfaţa RS232 sau CENTRONICS), un asamblor şi un compilator C. Toate kiturile sunt însoţite de documentaţie. Câteva din modulele de evaluare sunt:

Motorola EVBU, EVB, EVM, EVS sunt echipate cu MC 68HC11.

Dallas Semiconductor DS5000TK sunt echipate cu MC Dallas din seria DS5000.

Philips şi CEIBO DS750 sunt echipate cu 87C75x, echivalent cu 8051.

American Educational Systems AES-51(8051), AES-11(68HC11), AES-88(8088) conțin o tastatură, un afişaj cu LCD şi documentație.

Firma Texas Instruments pune la dispoziția celor interesați un set de accesorii pentru familia TMS370, un sistem de dezvoltare extins XDS şi unelte de dezvoltare (CDT). Aplicația poate fi scrisă în limbajul C, de exemplu pe un PC (utilizând orice editor) şi codul pentru TMS va fi generat de link editor. Codul poate ajunge la MC prin intermediul interfeței seriale RS232. XDS analizează softul creat, (chiar în domeniul timp) şi permite punerea la punct (chiar rularea lui pas cu pas).

Există firme specializate în construirea unor module de evaluare cu diferite MC, aşa cum este de exemplu PHYTEC (www.phytec.de).

Fiecare furnizor de MC pune la dispoziția clienților un Kit de start (Starter Kit), care este un instrument foarte valoros pentru a putea începe lucrul cu un anumit tip de MC. Kit-ul de start conține o placă cu soclu şi hard-ul aferent pentru programarea inițială a MC, uneori programarea EPROM-ului, interfața şi cablul de legătură la PC, documentație şi programe (asamblor, link editor, debugger) dar nu conține de regulă un MC. Prețul unui astfel de Kit porneşte de la 100USD (www.schuricht.de , www.farnell.com).

Un alt argument pentru alegerea unui MC este comoditatea folosirii lui. Unele MC, aşa cum este Motorola 68HC11A8P1 are înscris în ROM un program monitor (numit Buffalo). Pentru punerea în funcțiune este nevoie doar de o legătură serială RS232 cu un PC (este nevoie de o conversie de nivel) şi se poate lucra de pe PC cu monitorul MC, rulând programele în RAM-ul MC şi stocându-le în EEPROM. Pe Internet există subprograme scrise în Buffalo. Un alt MC, 8052AH-BASIC conține un BASIC. MC Dallas din seria DS5000 are nevoie doar de un cristal pentru oscilator şi este gata de lucru. MC conține memorie RAM static alimentată de la o baterie şi programul şi datele sunt astfel nevolatile.

6.2 ALGORITMUL PROIECTĂRII SISTEMELOR CU MC

Ne permitem să prezentăm o organigramă, (figura 6.1) ce încearcă să sugereze posibilele etapele pe care proiectantul de sistem ar fi recomandat să le urmărească în dezvoltarea soft-ului dedicat unei aplicații. Trebuie să menționăm că etapele 3,4 şi 5 pot fi iterate. Ele duc la perfecționarea funcționării sistemului.

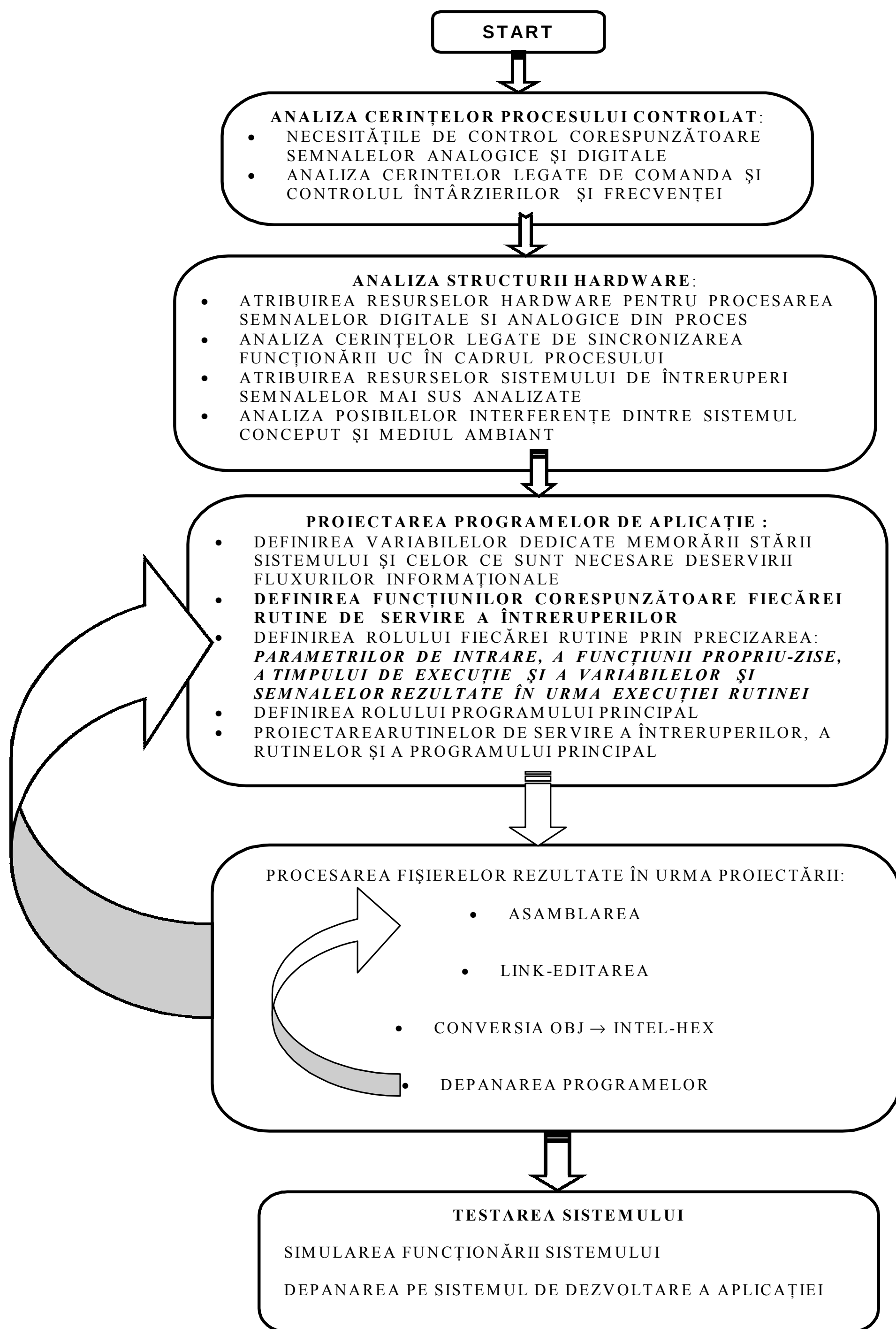


Figura 6.1 Organigrama (formă simplificată) proiectării unui sistem dedicat

6.3 PROIECTAREA SISTEMELOR CU MC ÎN VEDEREA SIGURANȚEI ÎN EXPLOATARE

Tendința producătorilor de MC de a scădea prețurile duce la răspândirea din ce în ce mai mare a MC şi la crearea a noi şi noi aplicații. Micşorarea dimensiunilor duce la creşterea frecvenței de lucru. Cu cât frecvența creşte, cu atât creşte posibilitatea interferențelor electromagnetice (EMI) şi se pune problema proiectării în vederea compatibilității electromagnetice (EMC).

Apar două categorii de probleme: aplicația poate genera perturbații (conduse sau radiate) sau poate fi susceptibilă la perturbații (conduse sau radiate). Descoperirea unor probleme de EMI în faza finală de producției a aplicației poate fi costisitoare deoarece s-ar putea să fie necesară reproiectarea aplicației; de aceea este necesar ca proiectarea inițială să se facă în vederea EMC.

Perturbațiile sunt generate de armonicile semnalelor digitale din circuit. Ele pot fi radiate de buclele de cablaj care se comportă ca şi antene sau sunt conduse spre sursa de alimentare. Orice cale inductivă sau capacitivă pe traseul acestor armonici poate provoca vârfuri de tensiune sau căderi de tensiune. Pentru un sistem cu MC perturbațiile sunt generate de regulă de cablaj, deoarece circuitele integrate au dimensiuni prea mici pentru a putea emite. Semnalul cu frecvența cea mai mare este tactul sistemului generat cu un circuit oscilant cu cuarț. Datorită faptului că forma semnalului este apropiată de forma sinusoidală, conținutul de armonici este mic. Dacă tactul este adus din exterior, se impune atenție mărită pentru a reduce buclele de circuit emisive.

Pentru un sistem care are memorii externe cuplate la MC, liniile de transfer pot fi emisive, deoarece frecvențele de tranziție sunt mari.

Susceptibilitatea sistemelor cu MC este creată de natura sincronă a MC. Un tact cu nivel electric insuficient poate produce o eroare. Erorile sistemelor cu MC pot fi grupate în:

1. aplicația are o eroare dar se corectează;
2. aplicația are o eroare dar o întrerupere sau un RESET corectează funcționarea;
3. aplicația are o eroare şi oprind şi repornind sistemul, eroarea dispare;
4. aplicația are o eroare permanentă datorată unei componente defecte.

Problemele din categoria 1 şi 2 pot să nici nu fie observate de beneficiar.

Performanțele EMC pot fi îmbunătățite acordând atenție deosebită următoarelor aspecte de proiectare:

- desenul cablajului imprimat;
- ceasul de gardă
- programarea defensivă

6.3.1 Cablajul imprimat

Orice perturbații pe liniile de alimentare pot produce o funcționare defectuoasă a sistemului. De aceea se recomandă utilizarea unui cablaj multistrat la care masa şi alimentarea sunt plane interne. La aplicațiile cu preț mic, o cale importantă de a reduce prețul este folosirea cablajului dublu placat sau chiar simplu placat. La aceste tipuri de cablaj traseele de masă şi +5V trebuie să fie cât mai late pentru a avea o impedanță cât mai mică. Decuplarea cu condensatoare a MC trebuie să fie realizată cât mai aproape de circuit.

În unele aplicații care trebuie să fie foarte ieftine se foloseşte alimentarea MC de la rețeaua de c.a. fără transformator, cu un redresor monoalternanță (figura 6.2).

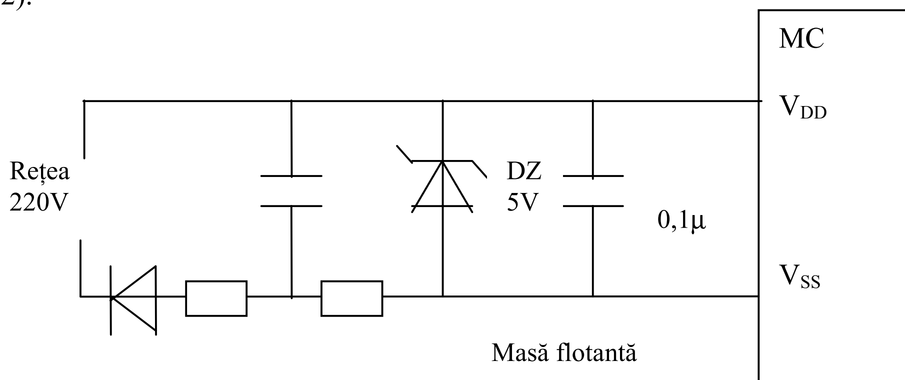


Figura 6.2 Schema simplă de alimentare de la rețea

În acest caz se creează o linie de mică impedanță spre masă prin rețeaua de alimentare de la V_{DD}. La acest tip de alimentare se recomadă decuplarea liniilor de frecvență înaltă spre linia de +5V.

Un circuit critic este cel de generare a tactulului. Orice impuls parazit care apare modifică factorul de umplere de 50% al semnalului de tact şi instrucțiunea nu se execută corect, circuitul ieşind din program. O astfel de situație singulară poate fi rezolvată de ceasul de gardă care comandă un RESET, după care MC rulează din nou corect. Se recomandă ca toate componentele aferente generării tactului să fie situate cât mai aproape de circuit, iar dacă este posibil toate traseele să fie înconjurate de un traseu lat de gardă. Se recomandă o decuplare a cristalului la linia de cea mai mică impedanță (de regulă masa) cu condensatori SMD. Se recomandă să se păstreze o distanță mare între liniile de frecvență mare şi circuitul de tact.

Este de asemenea importantă protecția pinilor de intrare, cum ar fi RESET sau IRQ. Un pin în aer poate comuta dacă în vecinătatea lui sunt linii de înaltă

frecvență. Se recomandă decuplarea acestor pini cu condensatori 1-10nF cât mai aproape de circuit.

6.3.2 Ceasul de gardă

Folosirea ceasului de gardă este foarte utilă în creşterea siguranței în funcționare. Majoritatea MC au un ceas de gardă integrat, iar dacă nu, un ceas de gardă extern poate fi uşor realizat cu un monostabil redeclanşabil.

Redeclanşarea ceasului de gardă trebuie făcută în programul principal, nu în subrutine. Pentru a putea folosi corect ceasul de gardă trebuie analizată cu atenție durata normală a programului pe ramura cea mai lungă.

6.3.3 Programarea defensivă

Prin metodele de programare defensivă se poate îmbunătăți mult siguranța în funcționare fără nici un hardware suplimentar. Câteva din cele mai eficiente metode sunt:

- reîncărcarea periodică a registrelor care comandă pinii I/O şi a celor mai importante registre. Pinii I/O sunt legătura MC cu exteriorul, de aceea ei sunt supuşi perturbațiilor. Readucerea lor la nivele corecte micşorează probabilitatea ca o perturbație să se propage în circuit.
- citirea repetată a semnalelor de intrare micşorează riscul unei citiri greşite. De exemplu citirea de 3 ori la rând a unui pin care este legat la o tastă. Dacă s-a citit aceeaşi valoare de fiecare dată se elimină posibilitatea unei perturbații.
- dacă există locații în RAM nefolosite, după fiecare etapă de rulare a programului se scrie un bit în RAM. Înainte de rularea unei rutine critice se verifică valoarea stocată în RAM şi rutina se execută doar în cazul în care valoarea din RAM este corectă.
- dacă într-o aplicație există memorie nefolosită, aceasta se umple cu instrucțiuni de salt într-un loc cunoscut pentru ca un salt neprevăzut în memorie datorat unei perturbații să fie anulat de saltul în locul cunoscut.
- Folosirea watchdog-ului în faza de testare şi chiar în aceea de fiabilizare a funcționării sistemului. Implică, postarea (memorarea într-o zonă specifică) stărilor succesive prin care trece sistemul, scrierea unei rutine de analiză a stărilor şi de directare a programelor către starea ultimă postată, rutină apelată la fiecare "warm reset", sau chiar

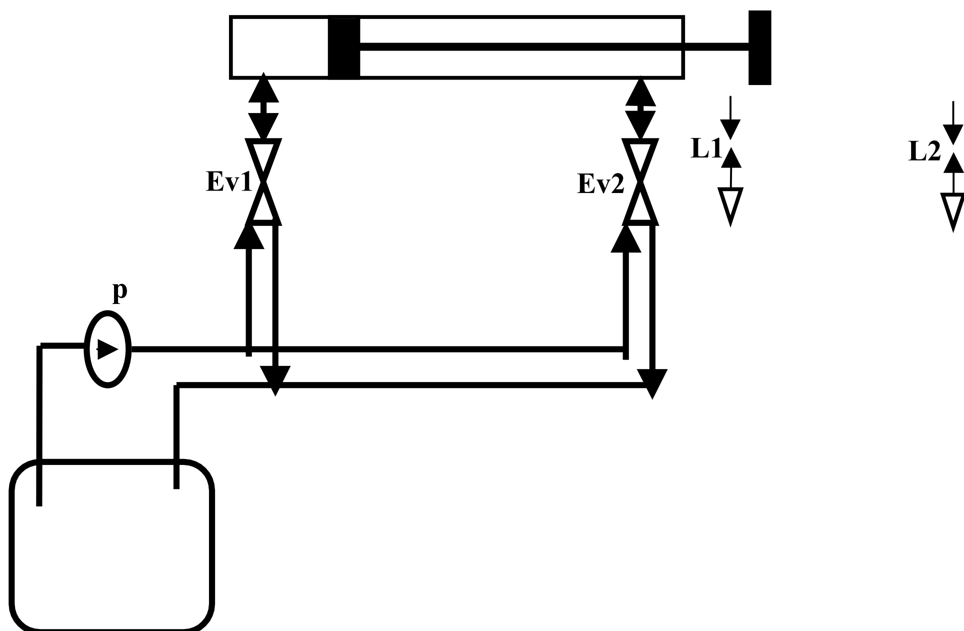
la fiecare reset al sistemului. Pot fi implementate soft şi alte funcţiuni care să permită menţinerea coerenţei sistemului!

7

APLICAȚII

7.1 IMPLEMENTAREA UNEI APLICAȚII SIMPLE DE COMANDĂ ŞI CONTROL DIGITAL

Să se comande utilizând un sistem dotat cu microcontroller un piston hidraulic (actuator) într-o mişcare de avans şi alta de retragere, comenzi transmise prin intermediul unei linii de conexiune seriale. Frecvența de transfer pe interfața serială va fi de 19200 bauds (biți pe secundă), iar frecvența de ceas a sistemului va fi de 11059000Hz. Cuvintele de comandă şi control sunt formate dintr-un singur caracter în format ASCII şi comenzile pot conține şi parametrii.



Cuvintele cheie admise de către sistem pentru controlul său sunt:

- A - comandă de avans a pistonului, respectiv comandă mişcarea de la limitatorul 1 către limitatorul 2
- R - comada de retragere a pistonului, respectiv comandă mişcarea de la limitatorul 2 către limitatorul 1
- S - comandă de citire a stării actuatorului care va întoarce următoarele ecouri:
    1. D - pregătit să recepționeze comandă (reaDy)
    2. B - ocupat cu execuția unei comenzi (Busy)
- P – comandă de aflare a poziției actuatorului (Position), care va întoarce următoarele ecouri:
    1. E – avansat, actuatorul atinge limitatorul 2
    2. I – retras, actuatorul atinge limitatorul 1
    3. M – actuator aflat în mişcare

Structura electro-hidraulică a sistemului de comandă şi control include elementele din figura 1. şi anume:

- pompa de presiune pentru circuitul hidraulic, care porneşte simultan cu comandă dorită; comanda pompei este realizată de către sistem în logică pozitivă (1 logic = pompa acționată, 0 logic = pompa oprită).
- două electro-valve cu două căi care asigură accesul presiunii către camerele pistonului hidraulic comandate tot în logică pozitivă, ceea ce înseamnă că aplicând un 1 logic pe linia de comandă, înalta presiune a pompei are acces către camera corespunzătoare a pistonului, iar aplicând un 0 logic pe linia de comandă presiunea din camera pistonului este eliberată către rezervorul de fluid.
- două limitatoare de capăt de cursă care funcționează în logică negativă şi care permit detectarea stării actuatorului.
- interfața serială RS232C cu care sistemul de comandă este conectat la sistemul ierarhic superior, spre exemplu la un PC.

7.1.1 Varianta de implementare cu microcontroller CISC

Proiectarea elementelor electronice ale sistemului o vom face utilizând circuitul AT89C2051, care este dotat cu 2 Ko memorie EEPROM (Flash) care permite stocarea programelor de comandă şi control. Sunt necesare atât amplificatoarele în comutaţie pentru comanda electrovalvelor şi a pompei de presiune cât şi circuitele de intrare ce se prezintă sub forma unor trigger-e Schmidt, pentru formarea semnalelor de la limitatoarele 1 şi 2. Structura gândită este prezentată în figura 7.2.

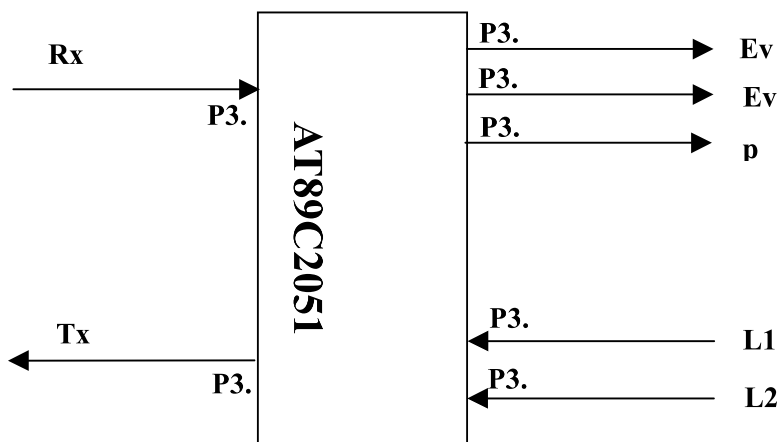


Figura 7. 2 Schema conectării microcontrolerului la aplicaţie

Implementarea software a sistemului implică scrierea programelor care să permită execuţia funcţiunilor sistemului propus. Ca sistem determinist, acesta trebuie să aibe o stare iniţială bine definită, de aceea vom considera că aceasta corespunde poziţiei cu pistonul retras aflat pe limitatorul L1, dar pe lângă aceste elemente ce ţin de procesul în care sistemul este utilizat, va trebui să analizăm şi să setăm parametrii iniţiali ai microcontroller-ului. Pentru aceasta, vom urmări starea iniţială a UC, prezentată în catalogul produsului (vezi figura 7.3), iar apoi va trebui ca să scriem rutina de iniţializare de sistem pentru a preciza complet starea microcontroller-ului.

Organizarea programelor care determină funcţiunile sistemului ilustrează principiile programării structurate, astfel: programul de iniţializare este apelat din programul principal, funcţiunile sistemului, cea de comunicaţie serială, cele de comandă avans şi retragere piston, cele de specificare a stării sistemului şi cele de oprire pe limitatoare a acestuia sunt specificate prin rutine de servire a întreruperilor (ISR –*Interrupt Service Routines*).

Numele SFR	Valori de RESET
PC	00H
ACC	00H
B	00H
PSW	00H
SP	07H
DPTR	00H
P0-P3	FFH
IP	xx000000B
IE	0xx00000B
TMOD	00H
SCON	00H
SBUF	Nedeterminat
PCON	0xxx0000B

Figura 7. 3 Valorile de RESET ale microcontroller-ului AT89C2051

1. Rutina de inițializare, INIT, va realiza urmatoarele:

- Umple toate locațiile memoriei interne a microcontroller-ului cu 0. (*Fill Memory with zero*)
- Aduce în pozitia de referință (pistonul atinge limitatorul L1) sistemul.
- Programează următoarele registre de funcții speciale (SFR *Special Functions Registers*)
  - PSW (*Program Status Word Register*)
  - PCON (*Power On Control Regsiter*)
  - TMOD (*Timers Mode Register*)
  - TCON (*Timers Control Register*)
  - SCON (*Serial Control Register*)
  - IE (*Interuppt Enable Register*)
- Intră în bucla de aşteptare a comenzilor de la sistemul ierarhic superior, comenzi ce vor fi transmise via interfața serială.

2. **Rutinele de serive a întreruperilor** vor fi concepute **pentru a implementa funcțiunile ce necesită sincronizarea funcționării sistemului cu elementele externe** acestuia ce sunt fie comandate fie reprezintă senzori sau traductoare ale sale. Astfel, va trebui să satisfacem următoarele:
   - Funcția de recepție a mesajelor de comandă sau a cererilor de date de stare din partea sistemului ierarhic superior. Implementarea acesteia se referă la interfața serială a sistemului şi necesită analiza mesajului receptionat, comanda corespunzătoare a sistemului sau emiterea ecoului la comenzile Cerere de Stare (*Status Request*) din partea sistemului ierarhic superior.

- ➢ Funcția de oprire la atingerea limitatoarelor a comenzii actuatorului electrico-hidraulic.

3. **Programul principal, este gândit să aştepte** o comadă din partea sistemului ierarhic superior sau să aştepte îndeplinirea acesteia.

OBSERVATIE: Această organizare a programelor poate fi utilizată mai ales atunci când sistemul controlat nu este unul extrem de rapid, respectiv noi, prin intermediul microcontroller-ului putem îndeplini restricțiile impuse unui sistem de comandă în timp real, adică putem oferi semnalele de comandă şi cele de control "ca oportune", respectiv ele verifică teorema eşantionării a lui Shannon.

În ceea ce însemnă implementarea, putem alege între două variante:

a) Cea mai sus menționată care permite ca scrierea funcțiunilor de comandă să fie facută direct în cadrul ISR
b) Cea prin care rutinele de servire a întreruperilor ISR consemnează modificările de stare în cadrul sistemului în zona dedicată variabilelor de stare, iar programul principal analizează ciclic starea sistemului, şi funcție de aceasta execută comenzile corespunzătoare ei.

Prima variantă, reflectă principiile programării structurate, fiecare ISR asigură implementarea unei funcțiuni. Această variantă permite cea mai rapidă reacție din partea sistemului, permite ierarhizarea reacțiilor în raport cu cerințele impuse de eşalonarea în timp a acestora, dar necesită o rațională atribuire a liniilor de întrerupere corespunzător evenimentelor principale din sistem şi este aplicabilă în special în cadrul unor sisteme de mică amploare.

A doua variantă asigură analiza stării sistemului în cadrul programului principal, analiza fiind implementată prin intermediul unor instrucțiuni de decizie, majoritatea calate pe bit, transferul informațiilor se face prin utilizarea unei zone comune de memorie ce formează legătura dintre lumea externă (elementele aflate în afara sistemului sau cele comandate: actuator, pompă, electrovalve, limitatoare, etc) şi funcțiunile pe care sistemul este obligat să le implementeze. Aceasta variantă permite controlul unor sisteme ample – cu multe surse de întrerupere şi multe elemente de comandă şi control- dar viteza de reacție este variabilă neputând fi anticipată aprioric, căci evenimentele ce au loc în afara sistemului vor fi consemnate sincron (adică foarte curând după producerea lor), dar analiza şi reacția sistemului la acestea va apare mai târziu, funcție de instrucțiunea curentă pe care sistemul o execută la momentul producerii evenimentului respectiv.

Să analizăm pe rând care vor fi informațiile pe care va trebui să le scriem în cadrul SFR pentru a inițializa sistemul.

PSW

CY	**AC**	**F0**	**RS1**	**RS0**	**OV**	**F1**	**P**

MSB LSB

CY *carry flag* este setat atunci când în urma unei instrucțiuni aritmetico-logice apare o operație de transport sau împrumut
AC *Auxiliary Flag* utilizat tot în cadrul unor instrucțiuni aritmetico-logice
F0 *User definible Flag* poate fi scris şi citit de către utilizator ca indicator specific
RS1 şi RS0 codifică natura (binar) bancul țintă de registre generale astfel:
00 specifică bancul 0 de registre generale
01 specifica bancul 1 de registre generale
10 specifica bancul 2 de registre generale
11 specifica bancul 3 de registre generale
OV este setat auotmat la apariția unei depăşiri în cadrul unei operații de transfer a datelor
F1 *User definible Flag* poate fi scris şi citit de către utilizator ca indicator specific
P *Parity Flag* specifică paritatea byte-ului de informație prelucrat

PCON

SMOD	**-**	**-**	**-**	**GF1**	**GF0**	**PD**	**IDL**

SMOD (*Serial Mode*) setat (dacă este fixat pe 1 logic) multiplică cu 2 frecvența de emisie/recepție serială a datelor
GF1 (*General Flag 1*), este un bit ce poate fi scris sau citit de către utilizator prin program
GF0 (*General Flag 0*), este un bit ce poate fi scris sau citit de către utilizator prin program
PD (*Power Down Flag*) Setarea acestui bit trece microcontroller-ul în starea de putere redusă blocând funcționarea oscilatorului intern şi aducând în starea high FFH ieşirile porturilor sale. Informațiile stocate în memoria internă a procesorului, abstracție făcând cele din registrele speciale (SFR) sunt menținute, chiar când tensiunea de alimentare a circuitului scade la 2 V. De asemenea, programul în curs va fi oprit imediat după efectuarea instrucțiunii curente în care a apărut setarea acestui flag. Acest flag are precedență în raport cu flag-ul IDL (Idle) atunci când cele două flag-uri sunt setate simultan.
Ieşirea din starea PD se face prin aplicarea unui semnal de Reset.
IDL (*IDLE Flag*) Setarea acestui flag duce la trecerea într-o stare de consum redus a microcontroller-ului, stare în care frecvența de ceas a sistemului este aplicată doar elementelor periferice, controller-ul de întreruperi, interfața serială, timer-ele etc. Porturile rămân în starea anterioară setării bitului IDL, iar semnalele ALE şi **PSEN#** trec în 1 logic. Revenirea în stare de lucru se poate realiza pe două căi:
Prin generarea externă a unei întreruperi, aceasta va reseta flag-ul IDL iar la terminarea ei, prin program, după instrucțiunea RETI putem seta din nou acest flag, sau printr-un Reset hardware, caz în care durata minimă a impulsului de Reset trebuie să depăşească 24 de ciclii de ceas.
Aceste două stări Power Down şi Idle Mode sunt stări în care consumul microcontroller-ului este redus şi pot fi exploatate atunci când aplicația pe care vrem să o realizam corespunde unui aparat portabil care trebuie să protejeze la maximum resursele energetice.

TMOD

GATE	C/T#	M1	M0	GATE	C/T#	M1	M0
Timer 1				**Timer0**			

Registrul TMOD nu poate fi accesat decât pe byte, deci scrierea unei informații în acest registru poate fi făcută prin instructiuni de tipul:

MOV TCON,#nnH sau
MOV @R0,#nnH unde R0=#TCON. Atenție, această instructiune poate scrie în memoria internă extinsă, dacă microcontroller-ul prezintă o astfel de memorie (vezi PCB80C552 şi alte analoge lui).
Modurile de funcționare ale canalelor temporizatoare/numărătoare sunt în număr de 4 şi sunt codate natural prin simpla scriere a numărului binar corespunzător canalului în biții 0 şi 1 pentru Timer0 sau 4 şi 5 pentru Timer1. Iată care sunt aceste moduri:
Modul 0, permite temporizarea[1] sau numărarea impulsurilor[2] aplicate la intrarea canalului respectiv dispunând de: un divizor cu 2^5 =32 (prescale) şi apoi divizarea programată cu o constantă de timp ce poate fi formată din orice număr reprezentabil pe 8 biți, deci cuprins între 0 şi 255, 0 corespunde unei divizări cu 256. Număratoarele canalului temporizator număra în sens direct, adică fiecare front negativ al unui impuls aplicat la intrare determină incrementarea numărătorului. Lățimea minimă a unui impuls este de 1/24 din frecvența de ceas a sistemului. Structura canalului temporizator numărător este dată în figura 4.
Modul 1, asigură temporizarea sau numărarea impulsurilor aplicate pe intrarea canalului, constanta de timp fiind un număr cuprins între 0 şi 65535 cu 0 corespunzător lui 65536.
Modul 2, permite divizarea frecvenței de intrare sau numărarea impulsurilor aplicate cu un număr reprezentabil pe 8 biti, aceasta constantă este reținuta în TH_x şi este reîncărcată automat în TL_x atunci când canalul numărător efectuează tranziția de stare din 11111111B în 00000000B.
Modul 3, este diferit la canalul 0 în raport cu canalul 1. Pentru canalul 0 setarea acestui mod permite împărțirea celor două numărătoare TL0 şi TH0 între pinii de comandă ai canalelor 0 şi 1. Temporizatoarele/numărătoarele canalului 0 low pot fi comandate de către pinii corespunzatori canalului 0 (TL0), şi temporizatoarele/numărătoarele canalului 0 high de către pinii corespunzători ai canalului 1 (TH0). Dacă programăm canalul 1 în modul 3 aceasta duce la blocarea acestuia şi reținerea informațiilor de la momentul respectiv. Utilizarea canalului 0

[1] Prin temporizare se înțelege măsurarea unui interval de timp ca multiplu al frecvenței de ceas a sistemului (microcontroller-ului). In acest caz impulsurile numărate de canalul respectiv provin de la ceasul de sistem, care în cazul circuitului AT89C2051 este divizat cu 12 şi aplicat pe intrarea de numărare.
[2] Numărarea impulsurilor ce sunt aplicate pe o intrare accesibilă din exterior (vezi pinii T0 sau T1). Aceste impulsuri pot proveni de la o sursă externă de semnal.

în modul 3 va permite multiplicarea canalelor temporizatoare/numărătoare cu un canal de temporizare/ numărătoare de 8 biți

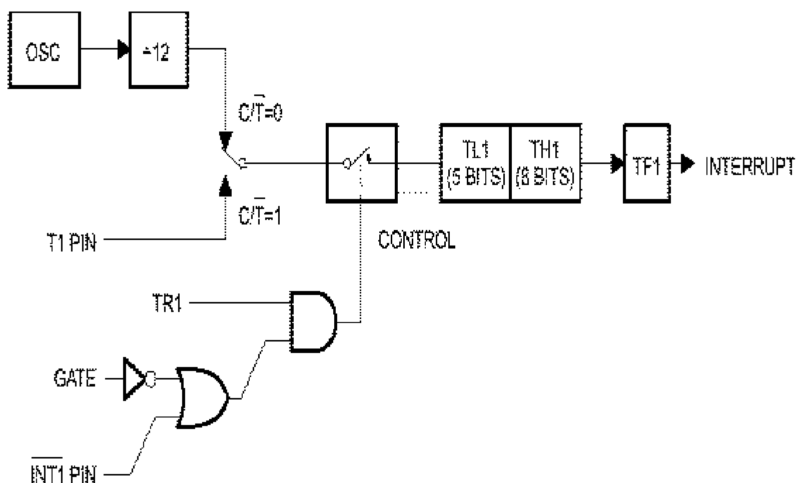


Figura 7. 4 Structura canalului temporizator al microcontrolerului AT89C2051

TCON

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Registrul TCON (*Timer Control*) permite modificarea setărilor bit cu bit, semnificația acestora o detaliem în continuare:

TF_x, exprimă starea canalului temporizator corespunzător. Setarea se realizează hardware, automat la trecerea de la starea 11…11B la starea 00..00B a numărătoarelor canalului respectiv (overflow).
Biții **TR_x** setați/resetați permit sau respectiv inhibă soft poarta de control ce asigură accesarea numărătoarelor de către semnalul de intrare al canalului respectiv.
Biții **IE_x** se setează automat (hardware) atunci când un front descrescător apare pe linia de întreruperi externe corespunzătoare şi sunt resetați tot automat la servirea întreruperii. (la intrarea în ISR).
Biții **IT_x** setați/resetați stabilesc dacă intrările corespunzătoare întreruperilor externe de stare vor fi sensibile pe frontul descrescător al semnalului, respectiv pe nivelul logic zero. În cazul în care biții respectivi (IT_x) sunt resetați, dacă semnalul extern aplicat pe intrările INT_x este de durată mai mare decât durata ISR corespunzătoare, la sfârşitul ISR se va genera o nouă întrerupere corespunzătoare respectivului canal.

SCON

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

SM0 şi SM1 setează unul dintre cele patru moduri distincte în care interfața UART a microcontroller-ului poate funcționa. Detalierea acestor moduri este realizată în continuare:

Modul 0, corespunzător combinației 00 a biților respectivi, permite serializarea octeților ce sunt transmişi şi receptionați semi-duplex pe linia RxD. Linia TxD este utilizată pentru transferul semnalului de ceas serial, în acest caz 1/12 din frecvența de ceas a microcontroller-ului.
Modul 1, corespunzător combinației 10 a biților respectivi, asigură transferul full-duplex asincron al informațiilor, respectiv 1 bit de start (0), 8 biți de date şi în final un bit de stop (1). Frecvența de transmisie este variabilă, ea fiind setată de către canalul 1 temporizator corespunzător frecvenței de apariție a stării overflow.
Modul 2, corespunzător combinației 01 a biților respectivi, asigură transferul asincron al informațiilor cu 1 bit de start (0), 8 biți de date, un bit programabil, care poate fi chiar bitul de paritate şi un bit de stop (1). La transmisie bitul 9 este reținut în locația TB8, iar la recepție acesta este memorat în RB8. Rata de transfer este programabilă la 1/32 sau 1/64 din frecvența de ceas a sistemului.
Modul 3, corespunzător combinației 11 a biților respectivi, asigură transferul asincron al informaiilor într-un format analog celui din modul 2 cu excepția faptului că rata de transfer este variabilă şi setată de către frecvența programată pentru canalul 1 al circuitului temporizator intern al microcontroller-ului.

Bitul **SM2** permite validarea unui mod special de comunicație via interfața serială USART şi anume a modului prin care pe acelaşi bus pot fi prezente mai multe sisteme care pot recepta informația. În acest caz, al 9-lea bit poate fi utilizat ca bit de specificare a tipului de informații ce sunt transmise. Astfel, setând bitul SM2, poate fi generată o întrerupere atunci când şi bitul al 9-lea transmis este 1 (spre exemplu), ceea ce va determina analiza de către sistemul receptor a informațiilor şi rejectarea acestora atunci când cuvântul de adresare respectiv nu corespunde propriei adrese.
Bitul **REN** permite validarea întreruperilor la recepție atunci când este setat, ceea ce face ca la recepția bitului de stop, corespunzător formatului de transmisie specificat, să se genereze o întrerupere şi totodată să se seteze bitul RI.
Bitul **TI** este setat harware la încheirea transferului unei informații via interfața serială, caz în care dacă întreruperile corespunzătoare acesteia au fost validate are loc transferul programului către ISR corespunzatoare. Resetarea acestui bit cade în responsabilitatea programatorului, acesta trebuind s-o efectueze în cadrul ISR.
Bitul **RI** semnalează recepția completă a unității de informație, via interfața UART. La recepția bitului de stop, flag-ul RI este automat (hardware) setat. Programatorul are obligația ca în ISR corespunzătoare recepției datelor să şteargă bitul RI, pentru a reanclaşa mecanismul de semnalizare a recepției datelor pe UART.

IE

EA	-	-	SI	TF1	IE1	TF0	IE0

Acest registru specifică sursele de întreruperi admise de microcontroller care vor putea genera întreruperi. Astfel:
IE0 corespunde liniei întreruperii externe de stare INT0
TF0 corespunde stării timer 0 overflow
IE1 corespunde liniei întreruperii externe de stare INT1
TF1 corespunde stării timer 1 overflow
SI corespunde întreruperilor ce provin de la unitatea UART integrată pe microcontroller.
Registrul IE permite validarea individuală a acestora, dar pentru ca o întrerupere să se declanşeze este necesar ca bitul EA să fie obligatoriu setat. Priorităţile în servirea întreruperilor sunt stabilite de către programator, fiind permise două nivele de prioritizare a întreruperilor: unul superior (high) setat prin înscrierea unui 1 logic în dreptul sursei de întrerupere respective (vezi structura pe surse de întreruperi a registrului IP - *Interuppt Priority Register*, este analogă celei a registrului IE), şi restul inferior (low). Sursa de prioritate înaltă va fi capabilă să întrerupă o rutină de servire ISR corespunzătoare unei surse de nivel de prioritate low, dar nu şi invers.
Dacă simultan apar mai multe întreruperi având acelaşi nivel de prioritate (low), ordinea de analiză a cererilor se face de la IE0 către SI.
Odată încheiată analiza structurii SFR să specificăm informaţiile adecvate aplicaţiei noastre. Iată-le:

In PSW vom scrie: **00000000B**.
In PCON vom scrie: **10000000B** setăm SMOD pentru multiplicarea cu doi a frecvenţei de emisie pe interfaţa serială.
In TMOD vom scrie: 00100000B, respectiv vom programa în modul 2 temporizatorul 1 cu autoreîncărcarea constantei de timp, iar canalul 0 îl vom menţine în modul 0.
Constanta de timp o vom calcula după ce vom seta şi parametrii interfeţei UART, respectiv vom transmite 8 biţi/caracter, la 19200 bauds, cu un bit de start şi unul de stop. În consecinţă în SCON vom scrie:

SCON : **01010000B**, adică vom alege modul 1 de funcţionare şi vom valida recepţia caracterelor pe interfaţa serială.

În acest caz, calculul constantei de timp se va face utilizând formula:

Baud Rate=$2^{SMOD}/32*f_{CLK}/(12*(256-TH_1))$. Rezultă, înlocuind în formulă valoarea de 253, respectiv **FDH, respectiv 11111101B** ce va trebui încărcată în TH1 drept constanta de timp.
În registrul IE vom scrie informaţia: **10010000B** corespunzătoare stării iniţiale a sistemului când acesta va trebui doar să accepte recepţia comenzilor de la sistemul ierarhic superior.

Iată rutina de inițializare a sistemului:

```
INIT:
                MOV   R0,#7FH
;Încarcă în registrul R0 limita superioară a memoriei interne
LOOP0:          MOV   @R0,#0H
;Scrie îndirect 0 în locația adresată prin pointer-ul R0
                DJNZ  R0,LOOP0
;Efectuează umplerea cu zero a memoriei interne a controller-ului
LOOP1:
                JNB   P3.2,START_POSITION
;Testare stare piston, Dacă atinge limitatorul L1, sistemul este în poziția de
;referință
                SETB  P3.6
                SETB  P3.5
                CLR   P3.7
;Comandă Electrovalva 1 şi pompa, şi anulează comanda pentru Electrovalva 2
                JMP   LOOP1
START_POSITION:
                SETB  STATUS0
                MOV   PSW,#00H
                MOV   PCON,#80H
                MOV   TMOD,#20H
                MOV   TH1,0FDH
                SETB  TR1
                MOV   SCON,#50H
                MOV   IE,#90H
;Setarea stării inițiale a microcontroller-ului implică validarea doar a întreruperilor
;UART
                RET
```

Rutina de servire a întreruperilor corespunzătoare UART va trebui să realizeze următoarele:

1. Să citească informația recepționată de pe linia serială.
2. Să analizeze informația, şi funcție de aceasta să reacționeze astfel:

- Dacă este vorba de o comandă să inițieze execuția acesteia.
- Dacă este vorba de o cerere de stare a microcontroller-ului din partea sistemului ierarhic superior să inițieze transferul acesteia.

Pentru a rezolva această funcțiune va trebui să introducem o variabilă locală de stare STATUS, având structura prezentată mai jos, actualizată de fiecare dată când are loc o schimbare a stării de către rutinele de servire a întreruperilor ce asigură sincronizarea funcționării unității centrale.

STATUS

-	-	-	-	-	CDA	L2	L1

- Bitul L1 este setat la atingerea limitatorului L1, în rest este resetat
- Bitul L2 este setat la atingerea limitatorului L2, în rest este resetat
- Bitul CDA este setat la inițierea unei comenzi de avans sau retragere şi resetat în rest, la sfârşitul mişcării.

Rutina de servire a întreruperilor din partea UART (ISR_SI) asigură tratarea fiecărui cuvânt de comandă dintre cele admise de protocolul instituit.

Locațiile STATUS asigură memorarea stării sistemului. în acest sens, asamblorul permite rezervarea de spațiu de memorie în zona de adrese 20H la 2FH, respectiv biții de la 00H la 7FH.

```
;Variabile de stare ale actuatorului electo-hidraulic.
STATUS0          BIT    20H
STATUS1          BIT    21H
STATUS2          BIT    22H
;Rutina de servire a întreruperilor generate de UART
;Atenție UART generează o unică întrerupere atât la recepție cât şi la transmisie
ISR_SI:
                 JB     RI,RECEPTIE
TRANSMISIE:
                 CLR    TI
;Şterge flag-ul Transmision Buffer Empty (Buffer de transmisie gol) pentru
;reanclanşarea întreruperilor la transferul unui nou caracter
                 JB     RI,RECEPTIE  ;Retestare recepție caracter de către UART
                 RETI
RECEPTIE:
                 MOV    A,SBUF     ;Citeşte caracterul recepționat în registrul ACC
                 CJNE   A,#'A',ISR_SI_1
CDA_AVANS:
                 SETB   P3.7    ;Comandă Electrovalva1
                 CLR    P3.6    ;Blochează Electrovalva2
                 SETB   P3.5    ;Comandă pompa de presiune
;Comandă mişcarea de avans a actuatorului acționând Electrovalva1 şi pompa.
                 CLR    STATUS0
;Resetează starea consemnată anterior, respectiv “actuator retras”
                 SETB   STATUS2           ;Consemnează comanda actuatorului
                 SETB   IE.1
;Validează şi întreruperile corespunzătoare limitatorului 2
                 JMP    END_ISR_SI
ISR_SI_1:
                 CJNE   A,#'R',ISR_SI_2
                 SETB   P3.6             ;Comandă Electrovalva2
                 CLR    P3.7             ;Blochează Electrovalva1
```

```
                SETB  P3.5             ;Comandă pompa de presiune
;Comandă mişcarea de retragere a actuatorului acționând Electrovalva 2 şi pompa.
                CRL   STATUS1
; Resetează starea consemnată anterior, respectiv "actuator avansat"
                SETB  STATUS2          ;Consemnează comandă actuatorului
                SETB  IE.0
;Validează şi întreruperile corespunzătoare limitatorului 1
                JMP   END_ISR_SI
;Testare cereri de stare din partea sistemului ierarhic superior
;Ecoul la acestea constă într-un caracter ce defineşte starea curentă a sistemului
ISR_SI_2:
                CJNE  A,#'S',ISR_SI_3
                JB    STATUS2,ISR_SI_4
                MOV   SBUF,#'D'
;Transmite ecoul corespunzător stării reaDy - pregătit
                JMP   END_ISR_SI
ISR_SI_3:
                CJNE  A,#'P',END_ISR_SI
;În cazul în care codul nu corespunde protocolului stabillit, - cuvântul recepționat
;nu este unul dintre cuvintele cheie- el va fi ignorat
                JB    STATUS0,LIM1
                JB    STATUS1,LIM2
                MOV   SBUF,#'M'
;Transmite ecoul "actuator în mişcare (poziție intermediară)
                JMP   END_ISR_SI
LIM1:
                MOV   SBUF,#'I'        ;Transmite ecoul "actuator retras"
                JMP   END_ISR_SI
LIM2:
                MOV   SBUF,#'E'        ;Transmite ecoul "actuator avansat"
END_ISR_SI:
                CLR   RI               ;Reseteaza flag-ul "recepție caracter"
                RETI
ISR_SI_4:
                MOV   SBUF,#'B'
                JMP   END_ISR_SI
```

Vom descrie în continuare rutinele de servire a întreruperilor ce asigură oprirea comenzilor de avans şi retragere a actuatorului electro-hidraulic. Corespunzător fiecărei linii de întreruperi externe vom scrie câte o rutină (ISR_EXT0 şi respectiv ISR_EXT1)

```
ISR_EXT0:
                CLR   IE.0
;Sterge flag-ul de validare a întreruperilor corespunzător sursei
```

```
                    CLR    P3.6           ;Şterge comada corespunzătoare lui Ev2
                    CLR    P3.5           ;Şterge comanda corespunzătoare pompei
                    CLR    STATUS2
;Şterge bitul ce consemnează mişcarea actuatorului
                    SETB   STATUS0
;Şetează bitul ce specifică starea "actuator retras"
                    RETI
ISR_EXT1:
                    CLR    IE.2
;Şterge flag-ul de validare a întreruperilor corespunzător sursei
                    CLR    P3.7           ;Şterge comada corespunzătoare lui Ev1
                    CLR    P3.5           ;Şterge comandă corespunzătoare pompei
                    CLR    STATUS2
;Şterge bitul ce consemnează mişcarea actuatorului
                    SETB   STATUS1
;Setează bitul ce specifică starea "actuator avansat"
                    RETI
```

Programul principal al sistemului va include doar o rutină de aşteptare a unui eveniment. Iată-l:

```
          ORG    0H
JMP       MAIN
          JMP    ISR_EXT0   ;Saltul la rutina de întreruperi corespunzătoare lui L1
          ORG    13H
          JMP    ISR_EXT1   ;Saltul la rutina de întreruperi corespunzătoare lui L2
          ORG    23H
          JMP    ISR_SI     ;Saltul la rutina de întreruperi corespunzătoare UART
          ORG    100H
$INCLUDE         INIT
$INCLUDE         SERIAL
$INCLUDE         EXT1&2
;Dispoziţii de inserare a programelor anterioare în cadrul programului principal
MAIN:
          CALL   INIT
;Construim tabela cu salturile la rutinele de servire a întreruperilor sau la programul principal
          JMP    $          ;Instrucţiunea asigură saltul permanent la ea însăşi
```

Urmare a scrierii programului, memoria de program a sistemului după parcurgerea etapelor de asamblare, link-editare şi conversie OBJ->HEX (IntelHEX code), va cuprinde în ordine, începând de la adresa 0 instrucţiunea de salt la programul principal, instrucţiunile de salt către rutinele de servire a întreruperilor şi rutinele de servire a întreruperilor şi programul principal în ordinea specificată prin dispoziţiile INCLUDE. Odată înscris în memoria flash a procesorului, sistemul este gata să satisfacă funcţiunile specificate în raport cu protocolul stabilit.

7.1.2 Varianta de implementare cu microcontroller RISC.

Implementarea aceleiaşi aplicații utilizând procesorul PIC16F84 presupune următoarele:

1. Implementarea interfeței cu aplicația, respectiv comanda celor două electrovalve şi a pompei de presiune
2. Implementarea controlului cursei pistonului, adică achiziția semnalelor digitale de la cele două limitatoare
3. Implementarea interfeței seriale ce asigură conexiunea către nivelul ierarhic superior, pentru aplicația noastră.

Vom analiza pe scurt care sunt principalele elemente caracteristice ale procesorului ales. Structura acestuia este prezentată în figura 7.5.

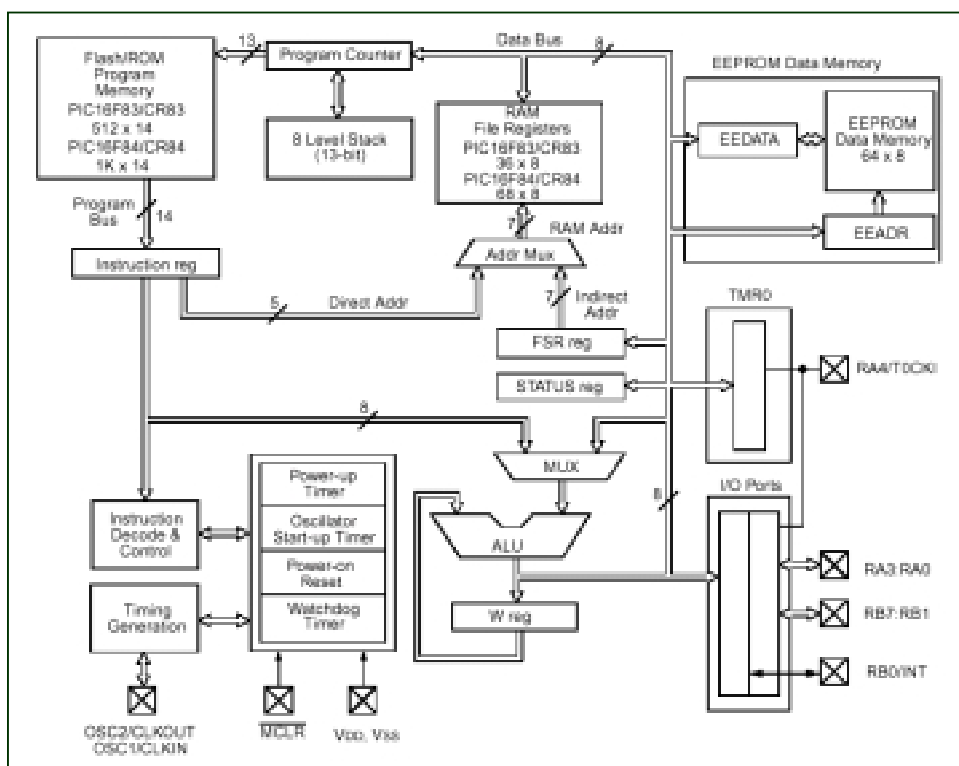


Figura 7. 5 Structura microcontroler-ului PIC16F86

Dispune de 13 linii utilizabile ca linii de intrare/ieşire cu capabilitate ridicată în curent (până la 20 mA, fiecare). Arhitectura procesorului este Harvard, cu lățimea magistralei de date de 8 biți şi a celei corespunzătoare memoriei de program de 14 biți. Dispune de 1Ko memorie de program (flash sau EPROM), de

2x128 bytes memorie SRAM organizată în două zone, zona SFR (*Special Function Regsiter)* şi zona GPR (*General Purpose Register*) şi de 64 bytes memorie EEPROM ce poate fi înscrisă şi stearsă prin program. Dispune de un canal temporizator/numărător ce poate îndeplini funcția de Watch Dog de 8 biți cu un prescaler de 5 biți. Controller-ul de întreruperi intern admite 4 surse de întreruperi: una externă (RB0/INT, una de la canalul TMR0 (timer overflow), una de la portul B comună pentru 4 dintre liniile portului care generează o întrerupere la schimbarea nivelului logic de semnal pe ele şi o întrerupere internă generată la scrierea datelor în EEPROM. Stiva microcontroller-ului este mai deosebită, respectiv este implementată într-o zonă specială a memoriei interne şi are maximum 8 nivele nefiind permisă citirea sau înscrierea acesteia prin adresare directă. Procesorul nu dispune de instrucțiuni PUSH sau POP şi astfel doar prin instrucțiuni de tipul CALL, GOTO, RETURN, RETFIE şi RETLW putem introduce, respectiv extrage informații la nivelul stivei. O observație importantă de care va trebui să ținem cont, este aceea că nu este prevăzut nici un indicator de depăşire a dimensiunii stivei, ceea ce poate duce la grave erori atunci când numărul de apeluri sau de instrucțiuni GOTO este mai mare de 8.

IND ADR	IND ADR
TMR0	OPTION
PCL	PLC
STATUS	STATUS
FSR	FSR
PORT A	TRIS A
PORT B	TRIS B
EEDATA	EECON1
EEADR	EECON2
PCLATCH	PCLATCH
INTCON	INTCON
68 x 8 GPR	Mapped

Figura 7. 6 Registrele microcontroler-ului PIC16F84

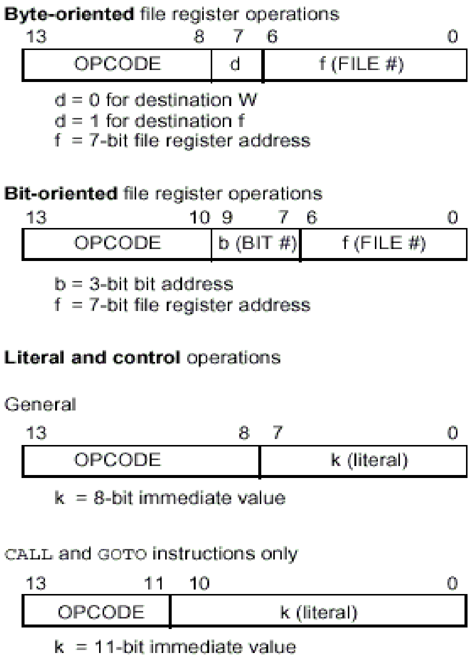


Figura 7. 7 Structura cuvântului instrucțiune

Instrucțiunile procesorului au o lățime de 14 biți şi ele includ atât codul respectiv cât şi operanzii aferenți acestuia, respectiv adresa

registrului din zona file Register şi/sau indexul bitului corespunzător locației respective.

Instrucțiunile de salt, cum ar fi, CALL şi GOTO sunt singurele care durează mai mult de un ciclu instrucțiune şi care necesită modificarea conținutului PC-ului.

Setul de instrucțiuni al procesorului este "ortogonal", ceea ce presupune că modalitatea de acces la orice locație de memorie este identică atât la scriere cât şi la citire, atât pentru zona GPR cât şi pentru zona SFR.

În cazul operațiilor calate pe bit, instrucțiunile vor citi întâi tot registrul, vor opera pe bitul selectat (specificat) şi vor întoarce înapoi rezultatul în registrul specificat.

Uniformitatea în tratare atât la nivel de bit cât şi la nivel de octet, precum şi a registrelor de uz general şi special şi a porturilor permite reducerea semnificativă a operațiilor de transfer intermediar care sunt specifice procesoarelor CISC.

Este de menționat că *Work Register* (**W** Register), funcționează ca acumulator şi participă la majoritatea instrucțiunilor atât cele de transfer direct cât şi la cele de transfer indirect –via un registru de adresare INDR -, adică un registru ce memorează adresa sursă sau destinație a informațiilor.

Iată câteva instrucțiuni pe care în cadrul programului le vom utiliza. Prezentarea lor o facem pentru că ele prezintă trăsături specifice în raport cu cele de la microcontroller-ul Atmel.

Aceste instrucțiuni le vom putea clasifica aşa în :

Instrucțiuni de transfer, aritmetico-logice şi de redirectare a programului, cum ar fi:

Pot conține drept parametri până la trei valori şi anume:

- Prin **k** (8 biți) specificăm o valoare imediată ce este inclusă în corpul instrucțiunii
- Prin **f** (8 biți) specificăm adresa din zona de memorie File Register la care face referire instrucțiunea.
- Prin **b** specificăm bitul la care face referire instrucțiunea (valoare pe 3 biți)
- Falg-ul **d** specifică registrul destinație în care se stochează rezultatul operației sau al transferului, dacă acesta este 0, atunci rezultatul este reținut în regsitrul W, altfel el este reținut în registrul f din File Register.

MOVF f,d Transfer data între W şi registrul f cu indicarea registrului de memorare a rezultatului.

MOVWF f Transfer data din W în registrul f

MOVLW k Transfer valoare imediată **k** (specificată în cadrul instrucțiunii) în registrul W.

SWAPF f,d Comutare semi-bytes în cadrul registrului f cu reținerea rezultatului în W sau f funcție de d.

ANDWF f,d SI-LOGIC, între informațiile din W şi f, bitul d are acelaşi rol,

ANDLW k SI-LOGIC între valoarea specificată k şi valoarea stocată în W. Rezultatul este reţinut în W.
ADDWF f,d Adunarea informaţiilor din W şi f, bitul d specifică destinaţia rezultatului.
ADDLW k Adunarea între valoarea specificată k şi valoarea stocată în W. Rezultatul este stocat în registrul W.
SUBWF f,d Scăderea informaţiile din W şi f cu acelaşi rol pentru flag-ul d
SUBLW k SI-LOGIC între valoarea specificată k şi valoarea stocată în W. Rezultatul este stocat în registrul W.
IORWF f,d SAU-LOGIC, între informaţiile din W şi f, bitul d are acelaşi rol,
IORLW k SAU-LOGIC între valoarea specificată k şi valoarea stocată în W. Rezultatul este reţinut în W.
XORWF f,d SAU-EXCLULSIV, între informaţiile din W şi f, bitul d are acelaşi rol,
XORLW k SAU-EXCLUSIV, între valoarea specificată k şi valoarea stocată în W. Rezultatul este reţinut în W.
CALL k Apel rutină **k** (11/13 biţi specificaţi prin instrucţiune)
GOTO k Salt la adresa **k** (11/13 biţi specificaţi prin instrucţiune)
RETURN Revenire din subrutină
RETFIE Revenire din subrutina de tratare a întreruperilor
RETLW k Revenire din subrutină cu încărcarea unei valori ”literale” în **W.**

Instrucţiunilede setare/resetare bytes sau biţi, cele de rotire pot fi încadrate în categoria curentă, unele dintre acestea le vom detalia, având în vedere particularităţile pe care ele le prezintă. Spre exemplu, instrucţiunea :

RRF f,d nu face altceva decât să deplaseze către dreapta cu o poziţie informaţia din registrul f, rezultatul fiind reţinut conform valorii lui d, în W sau chiar în registrul f. La fiecare deplasare, bitul 0 este mutat în bitul de C (de transport), iar acesta este transferat în bitul 7 al registrului f, etc.

Analog, funcţionează şi instrucţiunea **RLF f,d** , caz în care bitul C (de transport) va fi copiat în bitul 0 al registrului f şi bitul 7 al aceluiaşi registru fi memorat în C.

Instrucţiuni de decizie:

BTFSS Test bit şi salt peste instrucţiunea următoare dacă acesta este 1 logic
BTFSC Test bit şi salt peste instrucţiunea următoare dacă acesta este 0 logic

Instrucţiuni ce permit execuţia repetitivă a unui şir de operaţii:

DECFSZ f,d Decrementează registrul f şi sare instrucţiunea următoare dacă este zero rezultatul. Valoarea rezultat este reţinută funcţie de flag-ul d în W d=0 sau f d=1.

INCFSZ f,d Incrementează registrul f şi sare instrucțiunea următoare dacă este zero rezultatul. Valoarea rezultatului este reținută funcție de flag-ul d în W d=0 sau f d=1.

Restul instrucțiunilor de resetare sau setare pe bit şi byte, cele de incrementare şi decrementare, cele de complementare şi instrucțiunea NOP sunt furnizate pe CD. (sau vezi documentul 30430c.pdf de la Microchip).

Vom utiliza instrucțiunile de deplasare la dreapta prin bitul de carry pentru emularea UART.

Registrele generale ale microcontroller-ului asigură pe de-o parte implementarea funcțiunilor de control pentru program, iar pe de altă parte permit memorarea datelor şi transferul comenzilor către mediul extern. Iată structura celor mai importante dintre acestea.

STATUS

IRP	RP1	RP0	T0#	PD#	Z	DC	C

IRP este bitul care selectează bancul de registre (la adresarea indirectă) valoarea 0 corespunde intervalului de adrese 00H la FFH, iar valoarea 1 corespunde intervalului 100H la 1FFH.

RP1,RP0 corespunde selecției bancului de registre (la adresarea directă) selecție astfel:

00 corespunde intervalului de adrese 00H la 7FH

01 corespunde intervalului de adrese 80H la FFH

10 corespunde intervalului de adrese 100H la 17FH

11 corespunde intervalului de adrese 180H la 1FFH

T0# (*time out bit*), exprimă depăşirea capacității de numărare a canalului 0, respectiv tranziția de stare de la FFH spre 00H.

PD# (*Power Down*), odată setat exprimă "trezirea sistemului", la pornirea acestuia sau după execuția instrucțiunii CLRWDT. În rest bitul de mai sus este resetat.

Z (*Zero flag*), este setat la întâlnirea coincidenței a două valori numerice sau când rezultatul operației aritmetico-logice este zero. În rest, acesta este resetat.

DC (*Digital Carry*), exprimă transportul sau împrumutul la nivel de semibyte. Setat la apariția transportului, resetat la apariția unui împrumut.

C (*Carry*), este setat la apariția unui transport la nivel de byte, şi resetat în rest.

OPTION

RBPU#	INTEDG	TOCS	TOSE	PSA	PS2	PS1	PS0

RBPU#, (*Pull up bits port B*),setat exprimă invalidarea funcțiunii pull-up pentru liniile portului B, iar resetat, asigură funcția pull-up pentru aceste linii.

INTEDG (*Interrupt Edge Select*) setat permite generarea întreruperilor pe frontul crescător, iar resetat realizează generarea întreruperii pe frontul descrescător.

TOCS (*TMR0 Source Clock*), selectează sursa impulsurilor ce sunt numărate de către canalul 0 numărărtor/temporizator, setat permite numărarea impulsurilor externe, aplicate la pinul RA4, resetat selectează frecvența internă de ceas divizată cu 4.

TOSE (*TMR0 Source Edge Select*), setat realizează incrementarea numărătorului canalului pe frontul negativ, iar resetat incrementează impulsurile pe frontul pozitiv.
PSA, (*Prescaler Assignemet Bit*), setat asignează prescaler-ul pentru WDT (*Watch Dog Timer*), resetat îl rezervă pentru canalul TMR0.
PS2, PS1, PS0 (Biți de selecție a constantei de divizare a prescaler-ului), după cum urmează:
000H reprezintă 1/2 ptr. TMR0 şi 1/1 pentru WDT
001H reprezintă 1/4 ptr. TMR0 şi 1/2 pentru WDT
010H reprezintă 1/8 ptr. TMR0 şi 1/4 pentru WDT
011H reprezintă 1/16ptr. TMR0 şi 1/8 pentru WDT
100H reprezintă 1/32 ptr. TMR0 şi 1/16 pentru WDT
101H reprezintă 1/64 ptr. TMR0 şi 1/32 pentru WDT
110H reprezintă 1/128 ptr. TMR0 şi 1/64 pentru WDT
111H reprezintă 1/256 ptr. TMR0 şi 1/128 pentru WDT

Structura canalului temporizator/numărător este ilustrată în figura 7.8.

Întreruperile în cazul microcontroller-ului PIC16F84 au un singur "vector", şi anume cel plasat la adresa 004H din memoria de program, iar cum sursele de întreruperi sunt în număr de patru este necesar să implementăm în cadrul rutinei de servire programul de identificare şi apoi de tratare a acestora. Registrul

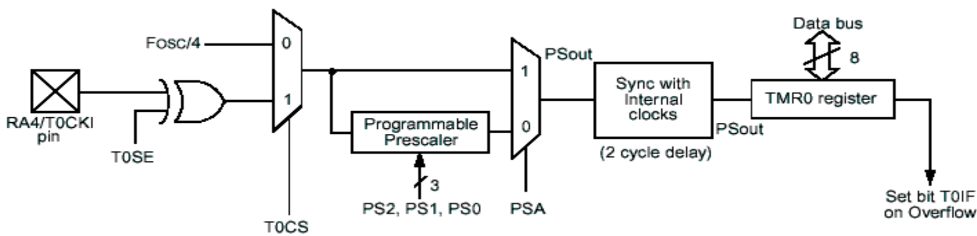


Figura 7. 8 Structura canalului temporizator la circuitul PIC16F84

INTCON, este cel ce dă posibilitatea utilizatorului să valideze sau invalideze atât declaşarea întreruperilor de la diverse surse, cât şi să valideze sau invalideze global întreruperile.

INTCOM

GIE	EEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF

GIE, (*Global Interrupt Enable*), permite setat să valideze acele cereri de întrerupere ce apar dacă acestea au fost validate şi individual, respectiv resetat inhibă orice cerere de întrerupere.
EEIE (*EEPROM Interrupt Enable*) setat validează întreruperile la operațiile de scriere sau citire a EEPROM-ului intern, resetat ignoră acestecereri de întrerupere
TOIE (*TMR0 Overflow Enable*), setat validează întreruperile corespunzătoare depăşirii la canalul temporizator/numărător, resetat acestea sunt ignorate

INTE (*External Interrupt Enable*), validează întreruperile datorate variației semnalului aplicat pe intrarea RB0 atunci când este setat, respectiv resetat le ignoră
RBIE (*B Register Interrupt Enable*), setat validează întreruperile datorate schimbării stării uneia dintre intrările portului RB7, RB6, RB5 sau RB4 al microcontroller-ului, respectiv, resetat ignoră schimbările de stare la nivelul portului RB
TOIF (*Timer Overflow Interrupt Flag*), este setat hard la apariția stării "depăşire" a capacității numărătorului corespunzător canalului 0, resetat în rest.
INTF (*Interrupt Flag*) setat hard la apariția unei întreruperi la nivelul liniei RB0, în rest resetat.
RBIF (*B Port Interrupt Flag*), setat hard, atunci când una sau mai multe linii ale portului B au schimbat starea, resetat în rest.

Ca observație generală trebuie reținut că biții de stare prezenți în structura registrului INTCON sunt setați hard (prin mecanismul implementat în cadrul microcontroller-ului) şi programatorul, în cadrul rutinei de servire a întreruperilor, va trebui să-i reseteze, pentru a putea detecta următoarea condiție de declaşare a întreruperilor.

Registrele EEDATA (*EEPROM Data Register*), reține data de înscris sau de şters din/în EEPROM, EEADR (*EEPROM address*), reține adresa de memorie la care, sau de la care se face: scrierea, respectiv citirea datelor şi registrele EECON1 şi EECON2 permit efectuarea memorării datelor în memoria EEPROM asignând comenzile de scriere WR sau citire şi cele de validare.

Având imaginea registrelor procesorului să gândim cum am putea utiliza acest microcontroller pentru transferul datelor şi implementarea funcțiunilor actuatorului electro-hidraulic.

Observăm un fapt mai puțin îmbucurător: circuitul nu dispune de o interfață serială implementată hard, deci va trebui să o emulăm soft.

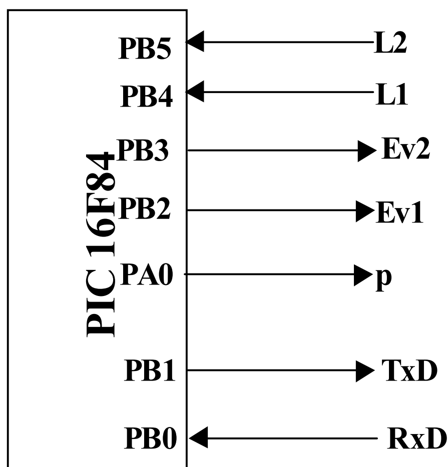


Figura 7. 9 Schema de conectare

În ceea ce înseamnă implementarea controlului, observăm că portul B este deosebit de util el detectând automat variația nivelului logic pe liniile RB4 la RB7. În figura 7.9 dăm schema electrică de conectare propusă, schemă în care am ignorat eventualele circuite amplificatoare în comutație sau formatoare de implus.

Pentru liniile RxD şi respectiv TxD, vom folosi liniile portului B şi anume RB6 (atenție vom utiliza şi facilitatea de generare a întreruperilor la schimbarea nivelului logic pe această intrare, în acest sens vom folosi posibilitatea pe care această linie o oferă,

de generare a întreruperii la sesizarea bitului de START în cazul transmisiei seriale), respectiv RB1 pentru transmisie.

Principalele idei în baza cărora vom emula interfaţa serială sunt următoarele:

- programăm timer0 astfel încât acesta să genereze câte o întrerupere cu frecvenţa cât mai apropiată de frecvenţa de transfer serial a informaţiilor.
- transferăm cu ocazia servirii întreruperii corespunzătoare, câte un bit şi rotim informaţia respectivă pentru a simula felul în care decurge transferul serial pe un UART implementat hard.
- vom stabili rata de transfer serială, programând corespunzător timer-ul 0 pentru aceasta. Iată cum se realizează acest lucru:

Formula de calcul -aproximativă[3]- pentru stabilirea frecvenţei de transfer, dacă presupunem utilizarea unui microcontroller ce funcţionează la f_{Clk}=10MHz va fi:

$$BR = \frac{f_{Clk}}{4} \cdot \frac{1}{PS \cdot (256 - CT)},$$

unde BR este rata de tranmisie serială (o vom stabili la 19200 Bauds), f_{Clk} este frecvenţa de ceas a sistemului, PS este factorul de divizare al prescaler-ului, iar CT este constanta de timp încărcată de către programator. Formula nu ţine cont de erorile datorate momentului de timp la care are loc reîncărcarea constantei de timp şi de timpul necesar microcontroller-ului pentru a intra în ISR.

În cazul nostru, vom obţine: 130 valoarea produsului PS*(256-CT).

Considerând pentru PS valoarea 1, deci nu utilizăm prescaler-ul, vom înscrie în registrul corespunzător timer-ului 0, valoarea 7EH.

Atenţie, este de observat că timer-ul numără în sens direct, deci valoarea corespunzătoare este dată de diferenţa între valoarea la care se realizează consemnarea depăşirii capacităţii canalului şi valoarea dorită.

Eroarea datorată rotunjirii valorii corespunzătoare ratei de transfer va fi de 0,16%.[4]

Pentru a implementa această funcţie, vom folosi câteva locaţii de memorie şi anume:

CHR_TRS: locaţie ce memorează octetul de transmis
CHR_REC: locaţie ce reţine octetul recepţionat
RI bit ce specifică recepţia completă a caracterului pe linia serială

---

[3] Formula este aproximativă căci ea nu ţine cont de timpii necesari execuţiei instrucţiunilor din cadrul rutinei de servire a întreruperilor, timpi care de altfel pot fi diferiţi, funcţie de starea în care este gasit procesorul la anclaşarea întreruperii

[4] Formula de calcul a erorii va fi: $Err = \frac{\tau_r - \tau_p}{\tau_p}$ unde τ_r este valoarea programată, iar τ_p este valoarea teoretică.

TI bit ce exprimă transmisia completă a caracterului pe linia serială
R bit ce exprimă transferul în curs la recepție (este utilizat de ISR).
T bit ce exprimă transferul în curs la transmisie(este utilizat de ISR).

Transferul se va efectua semi-duplex, cu 8 biți/caracter, un bit de start şi unul de stop. Structura byte-ului de stare este cea de mai jos:

SI_STARE

T	**R**	**TI**	**RI**	**END**	**C2**	**C1**	**C0**

Unde:

END are semnificația de sfârşit transmisie sau recepție byte

C2,C1,C0 sunt biții utilizați pentru contorul de biți, transmişi sau recepționați

Astfel, la transmisie vom face următoarele:

- Transferăm la locația CHR_TRS octetul de transmis şi setăm flag-ul INT_TRS.
- Resetăm linia TxD (respectiv bitul corespunzător ei, adică RB1) semnalizând bitul de start (Break pentru transmisie).
- Setăm bitul T, ce exprimă ocuparea CPU (*Central Processing Unit*) cu transferul serial.
- Inițiem ceasul de transmisie, adică programăm TMR0 în scopul generării întreruperilor
- Vom înscrie bitul corespunzător la fiecare întrerupere până la transferul complet al celor 8 biți.
- Setăm linia RB1, cel puțin pentru o perioadă de ceas, (bitul de stop), setăm bitul TI ce exprimă încheierea transmisiei caracterului ("buffer de transmisie gol") pentru programul principal şi resetăm bitul T - eliberând astfel UC de task-ul de serializare a informațiilor.

La recepție este necesar să parcurgem următoarele etape:

- La apariția schimbării de stare pe bitul RB0, setăm bitul ce consemnează ocuparea interfeței seriale, respectiv bitul R.("recepție în curs").
- Pe fiecare întrerupere de timp vom citi (eşantiona) linia serială la recepție – RB0 – şi vom scrie în Carry Flag bitul citit şi vom deplasa la dreapta prin carry byte-l CHR_REC ce va conține ceea ce receptăm.
- După 8 cicli de temporizare, vom seta bitul RI, ce specifică pentru programul principal, recepția completă a caracterului ("caracter recepționat disponibil"), resetând în acelaşi timp bitul CHR_REC. Va fi necesar să invalidăm întreruperile corespunzătoare canalului 0 temporizator, respectiv TMR0.

Având în vedere structura de registre ale procesorului, precum şi faptul că punctul de intrare în rutina de servire a întreruperilor este unic, va trebui ca aceasta

să implementeze absolut toate funcţiunile ce impun sincronizarea CPU cu mediul extern. De aceea, programul principal va iniţializa sistemul şi nu va face decât să aştepte întreruperile şi să dea comenzile corespunzătoare informaţiilor recepţionate din mediul extern după prealabila memorare şi analiză a acestora (informaţiile provin de la interfaţa serială sau sunt culese din proces).

Byte-ul de stare a sistemului va avea aceeaşi structură ca şi în cazul implementării anterioare. În plus, având în vedere variabilele care controlează starea interfeţei seriale UART, am mai introdus locaţia INT_TRS ce reţine cererea de transfer a unui caracter.

Iată în continuare programul ce implementează aplicaţia:

```
                list     p=16f84
                #include p16f84.inc
                ORG      0x00
START                    GOTO  MAIN
                ORG      0x04
ISR         ;Salvare stare microcontroller PIC, respectiv W şi STATUS
                MOVWF        TEMP_WORK    ;TEMP_WORK <-W
                MOVF  STATUS,W            ; W <- STATUS
                MOVWF        TEMP_WORK1   ;TEMP_WORK1 <- W
                BTFSC INTCON,INTF
;Testare apariţie bit de start pe RxD
                GOTO  PRIM_CHR
;Directare spre rutina ce iniţiază recepţia serială a caracterelor
                BTFSC INTCON,RBIF
;Testare modificare stare L1 sau L2
                GOTO  LIMITATORI
;Directare spre rutina de tratare corespunzătoare atingerii limitatorilor de cursă
                BTFSC INTCON,T0IF  ;Testare depăşire contor TMR0
                GOTO  REC_TRS
;Directare spre rutina corespunzătoare depăşirii contorului de timp corespunzător
                MOVF  INTCON,W     ; W <- INTCON
                ANDLW        0xF8          ;Resetez contorul de biţi
                MOVF  INTCON,F     ; INTCON <- W
END_ISR                            ;Refac starea microcontroller-ului
                MOVF  TEMP_WORK1,W
                MOVWF        STATUS
                MOVF  TEMP_WORK,W
                RETFIE
PRIM_CHR
                BCF   INTCON,INTE
;Invalidez INTE Nu accept alte întreruperi de la RxD. Revalidarea se va face doar
;la sfârşitul recepţiei complete a caracterului curent, odată cu setarea flag-ului RI)
                MOVLW        3/2*CT
```

```
;Incarc în W: 3/2 din constanta de timp corespunzătoare perioadei de transmisie
;serială a datelor (vezi diagrama de timp)
```
;De ce? Căci eşantionarea liniei RxD se face cât mai aproape de mijlocul ;perioadei de ceas a UART.
```
                MOVF  TMR0,F                  ; TMR0 <- W
                BSF   SI_STARE,6
;Setez indicatorul "în recepție" respectiv flag-ul R
                MOVF  SI_STARE,W
;Încarc în W vectorul de stare: W <- SI_STARE
                ANDLW        0xF8
;Resetez biții 0,1 si 2, respectiv contorul de biți la recepție
                MOVF  SI_STARE,F    ; SI_STARE <- W
;Se poate introduce în acest punct testul de "overrun" la recepție
                BCF   INTCON,INTF
;Resetez flag-ul INTE corespunzător liniei RxD
                GOTO  END_ISR
LIMITATORI
                BCF   INTCON,RBIF
;Resetez flag-ul indicator al întreruperii datorate variației stării liniilor RB4-RB7
                BTFSS PORTB,4       ;Testare atingere limitator L1?
;Da! A fost atins!
                GOTO  L1_ATINS
;Nu!
                BTFSS PORTB,5       ;Testare atingere limitator L2?
;Da! A fost atins!
                GOTO  L2_ATINS
;Ignorare întrerupere la schimbarea de stare, respectiv ignorare fronturi pozitive
;ale semnalelor de la limitatoarele 1 şi 2
                GOTO  END_ISR
L1_ATINS
                BCF   PORTB,3       ;Opreşte comanda Ev2
                BCF   PORTA,0
;Opreşte comanda pompa de fluid.   Actualizare stare sistem în vectorul STARE
                BSF   STARE,0       ;Atins limitator L1 poziție retras
                BCF   STARE,2       ;Oprită mişcare actuator
                GOTO  END_ISR
L2_ATINS
                BCF   PORTB,2       ;Opreşte comanda Ev1
                BCF   PORTA,0
;Opreşte comanda pompa de fluid Actualizare stare sistem în vectorul STARE
                BSF   STARE,1       ;Atins limitator L2 poziție avansat
                BCF   STARE,2       ;Oprită mişcarea actuatorului
                GOTO  END_ISR
REC_TRS
```

```
                BTFSC SI_STARE,6
;Test întrerupere de la TMR0 la recepţia caracterelor?
;Da! Intrerupere la recepţia caracterelor
                GOTO  RECEPTIE
;Nu!
                BTFSC SI_STARE,7
;Test întrerupere de la TMR0 la transmisia caracterelor?
;Da! Intrerupere la transmisia caracterelor
                GOTO  TRANSMISIE
;Nu! Eroare, refacem doar flag ce semnalează întreruperea coresp.  lui TMR0
                BCF   INTCON,T0IF
                GOTO  END_ISR
RECEPTIE
                BTFSS PORTB,0
;Eşantionare linie RxD (linia RxD este linia RB0) ;Da, linia este zero!
                GOTO  REC_ZERO
;Nu, Setez carry flag
                BSF   STATUS,C
;Setez bitul 0 din STATUS register care este Carry Flag
                GOTO  REC_CONT
REC_ZERO
                BCF   STATUS,C
;Resetez bitul 0 din STATUS register care este Carry Flag
REC_CONT
                RRF   CHR_REC,F
;Rotesc prin Carry locaţia (file register REC_CHR) Bitul de Carry
; va ajunge în poziţia bitului 7 din REC_CHR şi după 8 biţi receptaţi pe poziţia 0
                INCF  SI_STARE,F
;Incrementez contorul din SI_STARE şi îl salvez în el însuşi
                BTFSS SI_STARE,3
;Testez atingerea recepţiei celui de-al 8-lea bit al caracterului
                GOTO  END_REC_BIT
END_REC_CHR
                BCF   SI_STARE,6
;Resetez flag-ul ce indică starea "în recepţie caracter"
                BSF   SI_STARE,4
;Setez flag-ul RI, "Caracter recepţionat disponibil"
                BCF   INTCON,T0IF
;Resetez flag-ul ce semnalizează "overflow" canal temporizator
                BCF   INTCON,T0IE
;Invalidez întreruperile corespunzătoare canalului temporizator
                BCF   SI_STARE,3
;Resetez contor numărător biţi recepţionaţi
                GOTO  END_ISR
```

```
END_REC_BIT
                MOVWF       CT
;Încarc constanta de temporizare în W
                MOVF  TMR0,F                ;O transfer către TMR0
                BCF   INTCON,T0IE
;Resetez flag-ul corespunzător întreruperilor de la TMR0 Reanclanşare TMR0
                GOTO  END_ISR
TRANSMISIE
                BTFSC SI_STARE,3
;Test corespunzător încheierii transferului datelor ce formează caracterul (8 biți)
                GOTO  TEST_END_TRS
CONT_TRS
                BTFSC CHR_TRS,0
                GOTO  SET_BIT_TRS
                BCF   PORTB,1          ;Scriu 0 logic pe linia TxD
                GOTO  TRS_BITI
SET_BIT_TRS
                BSF   PORTB,1          ;Scriu 1 logic pe linia TxD
TRS_BITI
                RRF   CHR_TRS,F
;Rotesc biții caracterului de transmis în bufferul de transmisie
                MOVLW       CT
                MOVF  TMR0,F
;Reîncarc constanta de timp corespunzătoare frecvenței de transmisie pe UART
                BCF   INTCON,T0IF
;Resetez flag-ul indicator timer "overflow"
                BSF   INTCON,T0IE
;Validez întreruperile ptr. canalul temporizator
                INCF  SI_STARE
;Incrementez contorul de biți la transmisie
                GOTO  END_ISR
TEST_END_TRS
                BTFSC SI_STARE,0
;Testez dacă este de transmis bitul de STOP
                GOTO  END_TRS
TRS_STOP_BIT
                BSF   PORTB,1
;Scriu 1 logic pe linia serială, corespunzător bitului de STOP
                GOTO  TRS_BITI
END_TRS
                BCF   INTCON,T0IE
;Invalidez întreruperilor corespunzătoare TMR0
                BCF   INTCON,T0IF
;Şterg flag-ul indicator de întrerupere
```

```
                MOVLW       0x76
;Maschez bitul T, şi biţii contorului de biţi ai UART
                ANDWF       SI_STARE,F
;Realizez resetarea biţilor 7,3 şi 0 din SI_STATUS
                BSF   SI_STARE,5
;Setez TI, indică "buffer transmisie gol"
                GOTO  END_ISR
INIT
                MOVLW       0x0C
                MOVWF       FSR
NEXT
                CLRF  INDF
                INCF  FSR
                BTFSC FSR,6
                GOTO  NEXT
                BTFSC FSR,4
                GOTO  NEXT
;Umple cu 0 memoria SRAM a microcontroller-ului de la adresa 0x0C la 0x4F
                BCF   STATUS,RP0   ;Select bank 0
                MOVLW       0x02
                MOVF  PORTB,F
                MOVLW       0x01
                MOVF  PORTA,F
                BSF   STATUS,RP1   ;Select bank 1
                MOVLW       0x31
                MOVWF       TRISB
                MOVLW       0x00
                MOVWF       TRISA
                BCF   STATUS,RP0   ;Iniţializează porturile A şi B
                RETURN
;*****************************************************
MAIN            CALL  INIT
LOOP            BTFSC SI_STARE,4
;Testare dacă a fost recepţionat un caracter pe UART
                CALL  EXEC_CDA
;Execuţie comanda venită pe UART de la PC
CONTINUE0
                BTFSC INT_TRS,0
;Testare dacă este de transmis un caracter via UART
                CALL  INIT_TRS
CONTINUE1
                GOTO  LOOP
EXEC_CDA
                MOVLW       0x41   ;Încarc comanda 'A' - avans
```

```
                  XORWF        CHR_REC,W
;Testez identitatea între caracterul recepționat şi 'A'
                  BTFSC STATUS,Z        ;Testez identitatea Z=1 IDENTIC
                  GOTO  AVANS           ;Execută avans
                  MOVLW        0x52     ;Încarc comanda 'R' - retragere
                  XORWF        CHR_REC,W
;Testez identitatea între caracterul receptionat şi 'r'
                  BTFSC STATUS,Z        ;Testez identitatea Z=1 IDENTIC
                  GOTO  RETRAGERE
                  MOVLW        0x53              ;Incarc comanda 'S' - stare
                  XORWF        CHR_REC,W
;Testez identitatea între caracterul recepționat şi 'S'
                  BTFSC STATUS,Z        ;Testez identitatea Z=1 IDENTIC
                  GOTO  REQ_STARE
                  MOVLW        0x50     ;Încarc comanda 'P' - poziție
                  XORWF        CHR_REC,W
;Testez identitatea între caracterul recepționat şi 'P'
                  BTFSC STATUS,Z        ;Testez identitatea Z=1 IDENTIC
                  GOTO  POZITIE
END_EXEC_CDA
                  BCF   SI_STARE,4
;Resetez flag corespunzător caracterului recepționat.Acesta a fost citit şi interpretat
                  RETURN
AVANS
                  BSF   PORTB,2         ;Comandă Ev1
                  BSF   PORTA,0         ;Comandă pompa
                  BCF   STARE,0         ;Resetez stare sistem L1 atins
                  BSF   STARE,2         ;Setez starea "actuator în mişcare"
                  GOTO  END_EXEC_CDA
RETRAGERE
                  BSF   PORTB,3         ;Comandă Ev1
                  BSF   PORTA,0         ;Comanda pompa
          BCF     STARE,1         ;Resetez staresistem L1 atins
                  BSF   STARE,2         ;Setez starea "actuator in mişcare"
                  GOTO  END_EXEC_CDA
REQ_STARE
                  BTFSS STARE,2         ;Testez atingere limitator L1
;Da! Consemnează stare şi inițiază emisie ecou
                  GOTO  READY
BUSY
                  MOVLW        0x42     ; W <-'B' Busy
                  MOVF  CHR_TRS,F       ; CHR_TRS <- W
                  BSF   INT_TRS,0       ;Inițiere transmisie
                  GOTO  END_EXEC_CDA
```

```
READY
                MOVLW       0x44    ; W <-'D' reaDy
                MOVF  CHR_TRS,F     ; CHR_TRS <- W
                BSF   INT_TRS,0     ;Inițiere transmisie
                GOTO  END_EXEC_CDA
POZITIE
                BTFSC STARE,0
                GOTO  LIM1
                BTFSC STARE,1
                GOTO  LIM2
                BTFSC STARE,2
                GOTO  MOVE
                GOTO  END_EXEC_CDA
LIM1
                MOVLW       0x49    ; W <-'I' retras (L1 Atins)
                MOVF  CHR_TRS,F     ; CHR_TRS <- W
                BSF   INT_TRS,0     ;Inițiere transmisie
                GOTO  END_EXEC_CDA
LIM2
                MOVLW       0x45    ; W <-'E' avansat (L2 Atins)
                MOVF  CHR_TRS,F     ; CHR_TRS <- W
                BSF   INT_TRS,0     ;Inițiere transmisie
                GOTO  END_EXEC_CDA
MOVE
                MOVLW       0x4C    ;W <-'M' in mişcare
          MOVF  CHR_TRS,F     ; CHR_TRS <- W
                BSF   INT_TRS,0     ;Inițiere transmisie
                GOTO  END_EXEC_CDA
INIT_TRS
                BTFSS SI_STARE,5    ;Testez dacă TI este activ,
;În acest caz nu am voie să inițiez o nouă transmisie, voi aştepta!
                GOTO  END_INIT_TRS
                BCF   PORTB,1       ;Scriu 0 logic pe linia TxD
                MOVLW       CT
                MOVF  TMR0,F        ;Încarc constanta de timp înTMR0
                BSF   SI_STARE,7    ;Set "transmisie în curs"
                BSF   INTCON,T0IE
;Validez întreruperile corespunzătoare canalului TMR0
END_INIT_TRS
                RETURN
STARE                 DB    0x00    ;Vectorul de stare al actuatorului
;Bit 0 L1 atins
;Bit 1 L2 atins
;Bit 2 în mişcare
```

```
SI_STARE        DB   0x00  ;Vectorul stare corespunzător UART
CHR_REC         DB   0x00  ;Locație ce reține caracterul recepționat
CHR_TRS         DB   0x00  ;Locație ce reține caracterul de transmis
TEMP_WORK       DB   0x00  ;Locație ce memorează W pe durata ISR
TEMP_WORK1      DB   0x00
;Locație ce memorează STATUS register pe durata ISR
INT_TRS         DB   0x00
;Locația memorează în bitul 0 cererea de transmisie a unui caracter.
CT              EQU  0x7E  ;Constanta de timp coresp. ceasului UART
```

În text toate etichetele la care se face referire sunt scrise cu caractere îngroşate. Pentru a avea imaginea modului în care assembler-ul plasează codul nostru, dăm pe CD şi partea referitoare la locatare din listing-ul rezultat în urma asamblării. Această listă este dată cu titlu informativ pentru a putea sesiza modalitatea în care sunt asignate automat locațiile de memorie, variabilelor.

În final vom face o scurtă analiză comparativă a soluțiilor avute în vedere la implementarea aplicației.

Criteriul	Atmel AT89C2051	PIC16F84
Procesorul:		
Arhitectura	Von Neumann	Harvard
Setul de instrucțiuni	CISC[5]	RISC[9]
Registre de uz general	4 bancuri a câte 8 registre fiecare	2 Bancuri “File Registres” a câte 68 de bytes fiecare
Memoria internă	128 bytes	Este organizată sub forma "File Registrers"
Număr intrări/ieşiri:	15 I/O	13 I/O
Linii de comandă şi control		
Canale temporizatoare		
	2 (4 moduri de programare)	1 (2 moduri de programare)
WatchDog	1 canal (8 biți) 256 valori	
	1 canal sincron/asincron	1 canal (8 biți) 16 valori
UART	Vectorizat[6]	
Sistemul de întreruperi	Dispune de 2 intrări	NU
Alte facilități speciale	analogice[7]	Cu unică adresă de servire
	2,7 la 5 V	NU
Tensiune alimentare	Flash Reprogramabil	
Programarea controller-ului		2 la 6 V
Memoria de program	2 Kbytes (CISC)	Programare flash tip
Facilități energetice	Low Power şi IDLE mode[8]	ICSP [10]
		1024 x 14 biți (RISC)
		PowerDown[11] şi IDLE mode

[5] Complex Set Instruction Computer, deci microcontroller cu un set vast de instrucțiuni
[6] Prin sistem de întreruperi vectorizat înțelegem capacitatea dispozitivului respectiv de a realiza automat, la servirea unei întreruperi saltul la o adresă specifică sursei de întreruperi

Facilități software: Assembler Compilaror C Link-editor Translator INTEL-HEX code Emulator Mediu integrat	 A51, SAM51, etc. Extrem de multe variante de asambloare, Compilatoare C51 şi altele L51 DA DA NU	 MASM macro-asamblor, integrabil în mediul MPLAB DA MPLINK DA DA DA **(MPLAB)**
Aspecte economice: Preț/bucată Optenabilitate: Seria produsului:	 7,24 DM (Hoepping Elektronik) DA Nu avem date precise	 8,03 DM (ASA Micros) DA Nu avem date precise
Grad de pregatire proiectant:calat pentru microcontroller-ul respectiv	Nu avem date precise	Nu avem date precise
Suport de dezvoltare[12]:	DA	DA
Strategia firmei:	NU	NU
Timpul preconizat pentru dezvoltarea aplicației	NU	NU

Ce concluzii putem trage în urma analizei acestui tablou cu caracteristici ale celor două procesoare şi nu numai?

Putem stabili factori de pondere corespunzători fiecărui parametru (aceşti factori trebuie precizați de către fiecare proiectant în parte) şi putem calcula coeficientul de eficiență al implementării produsului.

Un aspect deseori esențial îl constituie pregătirea proiectanților, experiența lor anterioară şi timpul în care se preconizează dezvoltarea aplicației. Rubricile la care nu se poate răspunde de la început (vezi cele ce conțin "NU") pot influența substanțial eficiența deciziei, căci, spre exemplu, pregătirea unui specialist în domeniu presupune timp şi bani suplimentari pe care nu întotdeauna îi avem. (vezi rubrica "strategia firmei").

[7] Intrările pot fi conectate intern ca intrări inversoare/neinversoare ale unui comparator analogic, ieşirea acestuia fiind accesibilă soft, ca bitul 6 al portului 3 al microcontroller-ului
[8] Mod de funcționare în care procesorul consumă puțin căci reduce tensiunea de alimentare pe liniile porturilor şi opreşte execuția instrucțiunilor. Acceptă întreruperile ca modalitate de ieşire din stare dată
[9] Reduced Set Instruction Computer, deci microcontroller cu set redus de instrucțiuni
[10] ICSP prescurtare a "In Circuit Serial Programming", este facilitatea prin care utilizând drept linii de conexiune liniile microcontroller-ului se poate realiza programarea memoriei flash a circuitului fără ajutorul unui programator specializat
[11] Mod de funcționare în care procesorul nu execută instrucțiuni şi se izolează electric în raportul cu procesul controlat. Ieşirea din starea dată se face doar printr-un RESET hard.
[12] Exprimă existența atât a sistemelor de dezvoltare, a resurselor de programare: asamblor, link-editor, translator, emulator, simulator cât şi a programelor de proiectare asistată a implementării.

Alte informații privind microcontroller-ele familiei PIC de la Microchip găsiți la următoarele adrese:
Don McKenzie: http://www.dontronics.com/dtlinks.html
Adam Davis: http://www.ubasics.com/adam/pic/piclinks.shtml
Sam Powell: http://come.to/thepicarchive
Brian Lane: http://www.nexuscomputing.com/~picarchive/
Richard Spencer: http://engmtasd.derby.ac.uk
Michael Covington: ftp://ftp.ai.uga.edu/pub/microcontrollers/pic/
Steve Walz: ftp://ftp.armory.com/pub/user/rstevew/PIC/DaveTait/
http://www.tinaja.com/pic500.html
Iar pentru familia I8051 la adresele:
www.8052.com
www.intel.com
www.atmel.com
www.philips.com
www.hitachi.com

7.2 IMPLEMENTAREA INTERFEȚELOR LA PROCES

În acest paragraf vom aborda câteva aplicații care presupun folosirea convertoarelor analog-digitale şi a celor digital-analoge în aplicații de comandă şi control.

În acest sens vom exemplifica interfațarea sistemelor cu următoarele tipuri de convertoare:

1. Convertoarele analog – digitale controlabile serial
2. Convertoarele analog – digitale controlabile paralel
3. Convertoarele analog – digitale complexe
4. Convertoarele digital – analoge controlabile serial
5. Convertoarele digital – analoge controlabile serial.

Vom prezenta pentru fiecare dintre aceste cazuri atât aspectele legate de implementarea hardware, inclusiv caracteristicile convertoarelor, cât şi aspectele ce țin de implementarea software.

7.1.1 Convertoare A/D controlabile serial

Pentru primul exemplu, am ales un convertor reprezentativ pentru domeniul achiziției de semnale din cadrul instalațiilor industriale, şi anume: ADS7822, convertor cu regsitru de aproximații succesive a cărui schemă o dăm în figura 7.10.

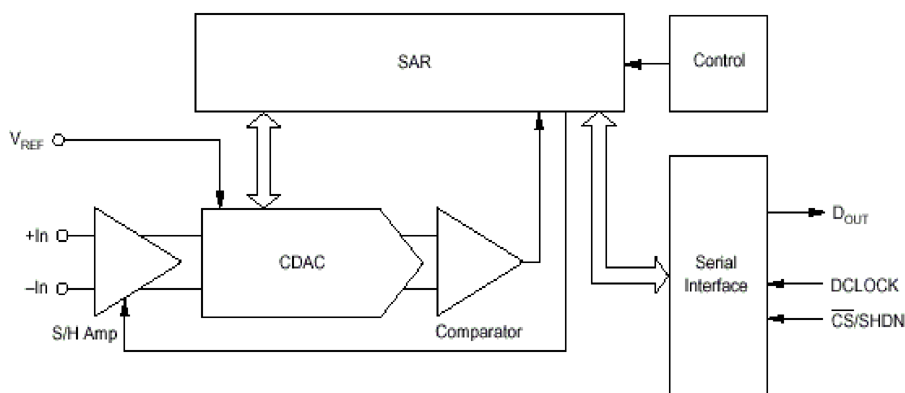


Figura 7. 10 Schema bloc a circuitului ADS7822 – convertor analog - digital cu registru de aproximații succesive

Convertorul prezintă un amplificator de instrumentație pe intrare la care ambele intrări sunt accesibile din exterior. Registrul cu aproximații succesive SAR (*Succesive Aproximation Register*), controlat de către semnalele de interfațare CS#/SHDN declanşează şi apoi controlează procesul de conversie, aplicând succesiv combinațiile ce implementează metoda divizării intervalului în aflarea valorii digitale corespunzătoare semnalului de pe intrare. Domeniul de măsurare este setat de către valoarea tensiunii sursei de referință externe ce aplică tensiunea stabilizată pe intrarea V_{REF}. Comparatorul întoarce la fiecare tact al convertorului rezultatul comparației şi astfel împreună cu logica de control realizează trecerea la testarea următorului bit. Convertorul va realiza în 12 ciclii testarea fiecărui bit de informație sincron cu semnalul de ceas DCLOCK, începând cu al 1,5-lea impuls de ceas (perioadă de eşantionare, caracteristică acestui convertor) după care informația va fi disponibilă bit de bit începând cu bitul cel mai semnificativ pe ieşirea D_{OUT} a circuitului. "Lățimea implusurilor" (durata acestora), nu trebuie să fie mai mică de 400ns, circuitul acceptând astfel frecvențe de ceas cuprinse între

10KHz şi 1,2 MHz, ceea ce corespunde unor rate de eşantionare a semnalului analogic cuprinse între:625 Hz şi respectiv 75kHz.

Impulsurile corespunzătoare de ceas sunt oferite de conexiunile pe care le vom realiza între convertor şi microcontroller. Am ales tot microcontroller-ul Atmel, având în vedere simplitatea schemei ce implementează aplicația şi faptul că acesta prezintă două canale numărătoare temporizatoare ceea ce favorizează aplicația. Timing-ul este prezentat în figura 11, iar schema de conectare în figura 12.

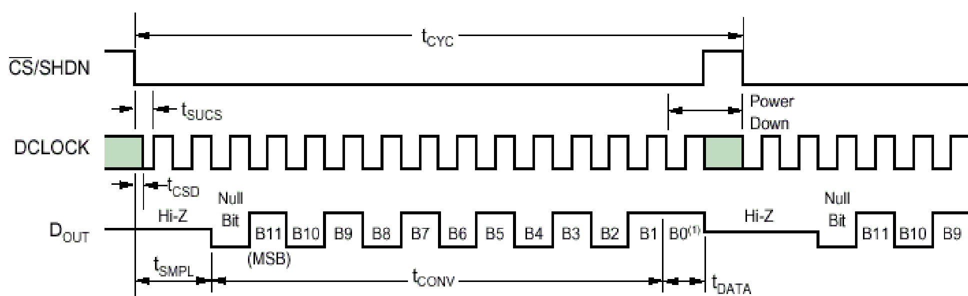


Figura 7. 11 Timing-ul (eşalonarea în timp) semnalelor de comandă pentru circuitul ADS7822

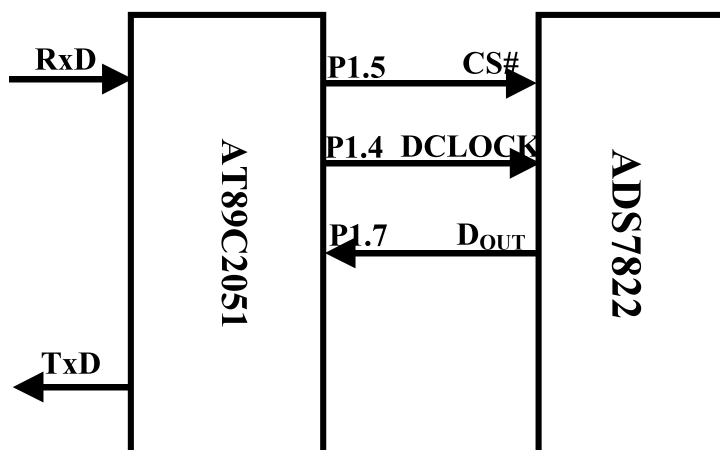


Figura 7. 12 Structura sistemului de achiziție – schemă simplificată

Vom utiliza canalul 1 al circuitului drept generator al ratei de transmisie serială a datelor, iar canalul 0 drept generator al frecvenței de ceas necesară convertorului A/D.

Formarea semnalului DCLOCK va fi realizată soft, prin generarea ratei de achiziție, programând corespunzător canalul respectiv. Aici este momentul să facem câteva comentarii.

Dacă microcontroller-ul funcționează la o frecvență de ceas de 24MHz, teoretic frecvența maximă pentru semnalul DCLOCK ar trebui să fie de 1MHz, dar va trebui să ținem cont de alte câteva "amănunte", şi anume:

- Întârzierea datorată procesului de servire a întreruperilor, care în acest care este de minimum 7 µs.
- De întârzierea datorată execuției instrucțiunilor rutinei de servire a întreruperilor
- De întârzierile (care pot fi aleatorii) datorate proceselor de schimb de informație concurente procesului de achiziție, cum ar fi procesul de transfer serial al datelor către sistemul ierarhic superior, alte procese ce concură la buna funcționare a sistemului, etc.

În final, vom specifica cum se calculează frecvența maximă de ceas corespunzătoare achiziției de semnal.

În figura 7.13 dăm structura principalelor fluxuri de informație în cadrul sistemului propus.

Proiectarea programelor în cazul astfel specificat, implică:

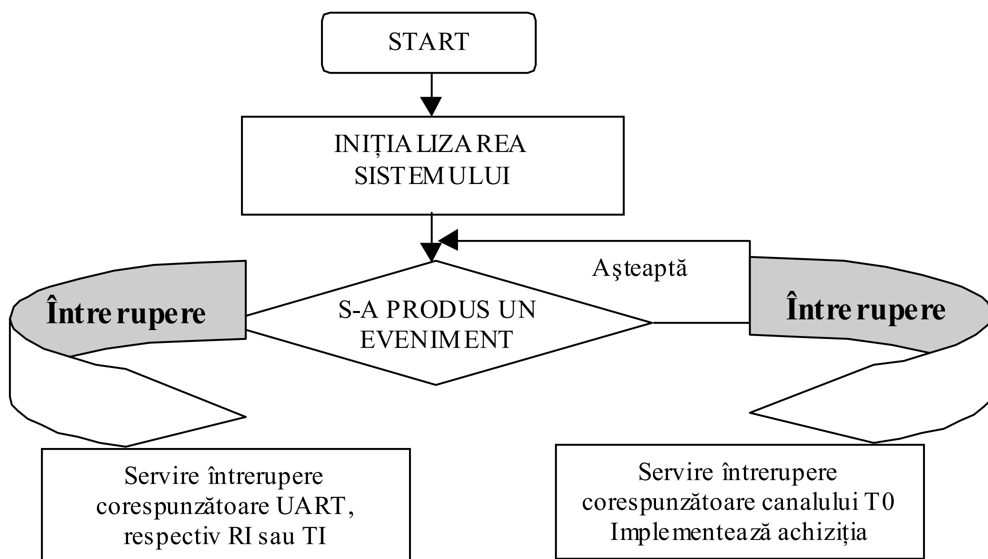


Figura 7. 13 Structura soft-ului aferent sistemului

Inițializarea de sistem, ceea ce presupune:

- Resetarea memoriei sistemului,
- Setarea valorilor corespunzătoare frecvenței de achiziție,
- Scrierea rutinelor de servire a întreruperilor
- Implementarea transferului semnalului de ceas către convertorul A/D
- Implementarea achiziției datelor de la convertor

- Implementarea buffer-elor de intrare/ieşire
- Implementarea transferului serial al datelor, la frecvenţa şi în condiţiile dorite.
- Scrierea programului principal, care are ca principal rol, rolul de a aştepta producerea unor evenimente.

Dăm în continuare programul ce implementează aplicaţia.

```
;Rutina de servire a întreruperilor corespunzătoare achiziţiei de semnal
;Realizează achiziţia unei date în cadrul ei, necesită următoarele locaţii:
CT_H        EQU   11H
CT_L        EQU   00H
SET_UART    EQU   5CH    ;Setarea parametrilor interfeţei seriale
SET_TIMER   EQU   21H
;Setarea modului de funcţionare pentru timer-ele microcontroller-ului
SET_T1            EQU    0FDH
;Setarea constantei de timp pentru rata de transfer a UART
            DSEG
            ORG   24H
STATUS      DATA  00H
CONTOR      DATA  00H    ;Locaţia memorează nr. de pasi la achiziţie
DATA_REC    DATA  00H    ;Locaţia memorează comanda venita via UART
DATA_LSB    DATA  00H    ;Locaţie byte mai puţin semnificativ
DATA_MSB    DATA  00H    ;Locaţie byte mai semnificativ
            BSEG
TRS_MSB           BIT    10H
;Specifică necesitatea transferului celui mai semnificativ byte al datei achiziţionate
REC_EN            BIT    11H
;Specifică recepţia completă a unui caracter, caz în care programul principal îl va
;analiza generând ecourile specifice comenzii primite
;Rutina de servire a întreruperilor corespunzătoare canalului 0 temporizator
;In cadrul programului principal se vor seta:
;rata de eşantionare prin setarea constantei de timp
;modul de achiziţie: 1.un singur punct / 2. puncte multiple
            CSEG
            ORG   0H
            JMP   MAIN
            ORG   0BH
ISR_T0:
      (24)  PUSH  PSW    ;Salvez contextul respectiv registrele PSW şi ACC
      (24)  PUSH  ACC
      (12)  SETB  PSW.3
;Schimb bancul ţinţă de registre generale cu bancul1
      (12)  MOV   TH0,#CT_H
      (12)  MOV   TL0,#CT_L
```

```
;Reîncarcă constantele de timp şi redeclaşează temporizarea corespunzătoare
        (12)    CLR     IE.1    ;Resetează flag-ul de întrerupere corespunzător
canalului 0
        (12)    SETB    P1.5
        (12)    CLR     P1.5    ;Activează CS# pentru accesul la convertor
        (12)    CLR     P1.4
;Resetează DCLOCK pentru 1 ms. Formează DCLOCK =0
        (12)    SETB    P1.5    ;Formează DCLOCK =1
        (12)    CLR     P1.5    ;Formeaza DCLOCK =0
;S-a încheiat perioada de eşantionare pentru convertor. Urmează citirea datelor
;Aceasta presupune următoarele: DCLOCK=1, Incrementează contor,DCLOCK=0,
;Citeşte data. Numărul de cicluri este de 13 având în vedere că prima valoare citită
;de la CAD este 0
        (12)    MOV     R0,#8   ;Initializez contor
        (12)    MOV     A,#00H;Resetez ACC
LOOP_MSB:
```

180/12=15µS este timpul cât durează execuția instrucțiunilor de mai sus

```
        (12)    SETB    P1.4    ;Formeaza DCLOCK =1
        (12)    CLR     P1.4    ;Formeaza DCLOCK =0
        (24)    JNB     P1.7,CLR_BIT8
        (12)    SETB    ACC.0
CLR_BIT8:
        (12)    RL      A
        (24)    DJNZ    R0,LOOP_MSB
        (12)    MOV     DATA_MSB,A ;Salvez biții 4 la 12 achiziționați
        (12)    MOV     R0,#4   ;Inițializez contor
        (12)    MOV     A,#00H;Resetez ACC
```

180/12=28,5µS este timpul cât durează execuția instrucțiunilor de mai sus

```
LOOP_LSB:
        (12)    SETB    P1.4    ;Formeaza DCLOCK =1
        (12)    CLR     P1.4    ;Formeaza DCLOCK =0
        (24)    JNB     P1.7,CLR_BIT0
                SETB    ACC.0
CLR_BIT0:
        (12)    RL      A
        (24)    DJNZ    R0,LOOP_LSB
        (12)    MOV     DATA_MSB,A ;Salvez biții 0 la 3 în DATA_LSB
;Am achiziționat toți cei 12 biți oferiți de ADS7822
```

180/12=14,5µS este timpul cât durează execuția instrucțiunilor de mai sus

```
        (12)    SETB    P1.5    ;Inactivez CS#
;Formez rezultatul conversiei
        (12)    MOV     A,DATA_MSB
        (12)    SWAP    A
        (24)    PUSH    ACC
```

```
        (12)   ANL    A,#0F0H              ;Maschez LSB=biții 4-7 ai LSB
        (12)   ORL    A,DATA_LSB
        (12)   MOV    DATA_LSB,A  ;Formez primii 8 biți ai datei convertite
        (24)   POP    ACC
        (12)   ANL    A,#0FH          ;Maschez cel mai seminificativ semibyte
        (12)   MOV    DATA_MSB,A ;Formez   următorii   4   biți   -cei   mai
;semnificativi
        (24)   POP    PSW    ;Refac starea PSW dinaintea intrării în întreruperi
        (24)   POP    ACC    ;Refac starea ACC dinaintea intrării în întreruperi
        (24)   RETI
```

180/12=8,5µS este timpul cât durează execuția instrucțiunilor de mai sus

66,5µS ESTE TIMPUL TOTAL CORESPUNZĂTOR ISR _T0

```
;Rutina de servire a întreruperilor corespunzătoare UART
;Sistemul va transmite date doar la cererea sistemului ierarhic superior şi
;numai maximum doi bytes
               ORG    23H
ISR_SI:
        (24)   JB     RI,REC ;Testare dacă a fost recepționat un caracter
TRS:                         ;NU! Intrerupere la transmisie
        (12)   CLR    TI
        (12)   JNB    TRS_MSB,END_TRS;Test dacă mai sunt de transmis date
        (12)   MOV    SBUF,DATA_MSB      ;Transfer MSB data achiziționată
        (12)   CLR    TRS_MSB
        END_TRS:
        (24)   RETI
REC:
        (12)   CLR    RI
        (12)   MOV    DATA_REC,SBUF      ;Scriu data în buffer-ul de recepție
        (12)   SETB   REC_EN             ;Specific recepția unui caracter
        (24)   RETI
MAIN:
               CALL   INIT              ;Rutina de initializarea a sistemului
LOOP:
               JNB    REC_EN,LOOP
ANALIZA:
               MOV    A,#'A'  ;Incarcă primul caracter utilizat drept comanda
               CJNE   A,DATA_REC,CONTINUE0
               CALL   INIT_ACHIZITIE
               JMP    LOOP
CONTINUE0:
;Aici pot fi inserate celelate teste pentru comenzile pe care le instituim prin
protocol
;… … …
               NOP
```

INIT_ACHIZITIE:

```
;Rutina de initiere a achizitiei datelor trebuie sa valideze intreruperile
;corespunzatoare canalului 0 si sa incarce constanta de timp corespunzatoare
;ratei de esantionare
                MOV   TH0,#CT_H
                MOV   TL0,#CT_L
                SETB  TR0
                SETB  T0
                RET
```

INIT:

```
;Rutina de inițiere de sistem. Nu mai detaliem inițializarea memoriei
                MOV   IE,#90H  ;Inițializez întreruperile corespunzătoare UART
                MOV   PSW,#0  ;Inițializez PSW
                MOV   SCON,#SET_UART      ;Inițializez UART
                MOV   TMOD,#SET_TIMER
;Setez modurile de funcționare ale timer-elor
                MOV   TL1,#0
                MOV   TH1,#SET_T1   ;Setez constantele de timp
                RET
```

Să observăm că dacă calculăm timpul de execuție corespunzător instrucțiunilor rutinei de servire a întreruperilor la achiziția datelor în cazul unei frecvențe de ceas de 24 MHz obținem o durată de 66,5 μs, adică frecvența maximă de achiziție este de: 15.037 Hz. Putem constata că şi în situația în care am utilizat facilitățile hard de sincronizare, respectiv sistemul de întreruperi totuşi viteza de achiziție este puternic legată de modul de implementare prin program a acesteia, deci acest aspect va trebui cu cea mai mare grijă analizat şi proiectat.

Observăm că dacă am fi utilizat la achiziția datelor de la convertorul analog-digital facilitatea interfeței UART a microcontroller-ului (vezi modul 0 de funcționare) am fi putut creşte frecvența maximă de achiziție aproape de limita superioară corespunzătoare convertorului analog-digital. În acest caz, observăm că ar fi fost necesar să recepționăm sincron doi octeți, ceea ce ar fi dus la consumarea unui timp de aproximativ 16 ciclii de ceas, adică la 24 MHz timpul de achiziție ar fi fost de 8μs, ceea ce corespunde unei rate de eşantionare de 125000Hz ceea ce evident că depăşeşte frecvența maximă admisibilă a convertorului. Evident, în acest caz am fi renunțat la utilizarea UART drept port de legătură cu sistemul ierarhic superior.

7.1.2 Interfațarea paralelă a convertoarelor A/D

În acest scop vom utiliza convertorul ADS7821 convertor cu registru de aproximații succesive de 16 biți. Câteva date tehnice importante: frecvența maximă 100 kHz, domeniul tensiunilor de intrare: 0 la 5 V, poate utiliza atât o referință de tensiune internă, cât şi una externă, poate fi conectat direct la un bus de sistem. Schema bloc a sa este prezentată în figura 7.14.

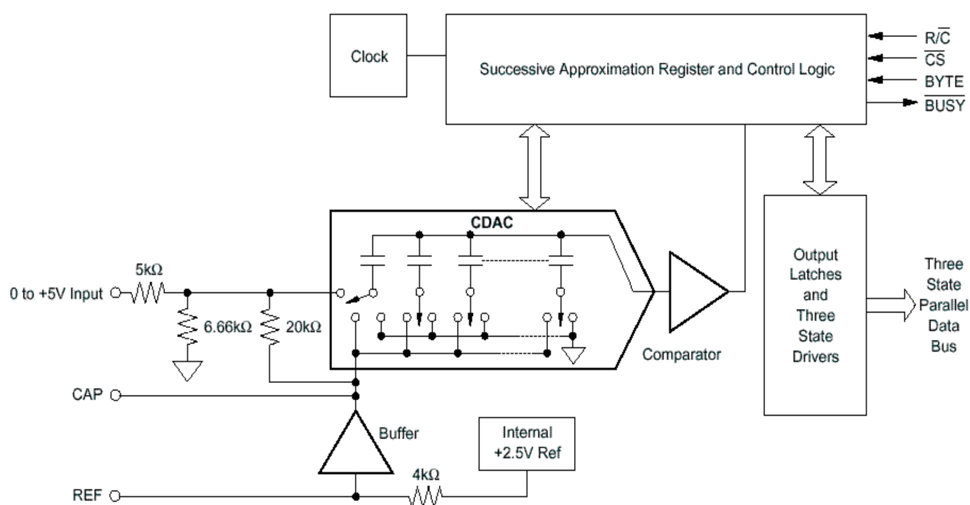


Figura 7. 14 Structura convertorului analog-digital ADS7821, cu ieşire paralelă de date

Pentru a-l putea conecta, este necesar să cunoaştem care este modul de funcționare al convertorului. Astfel, semnalul **Read/Convert (R/C#)** asigură citirea rezultatului sau declanşarea conversiei, semnalul **CS#** permite trecerea bus-ului din starea High Impedance într-una din stările 1 sau 0 logic, semnalul **BYTE** permută magistralele low/high de ieşire, iar linia **BUSY#** semnalează situația în care convertorul se găseşte în faza de eşantionare şi conversie a semnalului analogic.

Pentru a realiza conversia de semnal va trebui să cunoaştem care este succesiunea de semnale de comandă. Combinațiile posibile sunt prezentate în tabelul I.

$\overline{\text{CS}}$	R/$\overline{\text{C}}$	$\overline{\text{BUSY}}$	OPERATION
1	X	X	None. Databus is in Hi-Z state.
↓	0	1	Initiates conversion "n". Databus remains in Hi-Z state.
0	↓	1	Initiates conversion "n". Databus enters Hi-Z state.
0	1	↑	Conversion "n" completed. Valid data from conversion "n" on the databus.
↓	1	1	Enables databus with valid data from conversion "n".
↓	1	0	Enables databus with valid data from conversion "n-1"(1). Conversion n in process.
0	↑	0	Enables databus with valid data from conversion "n-1"(1). Conversion "n" in process.
0	0	↑	New conversion initiated without acquisition of a new signal. Data will be invalid. $\overline{\text{CS}}$ and/or R/$\overline{\text{C}}$ must be HIGH when $\overline{\text{BUSY}}$ goes HIGH.
X	X	0	New convert commands ignored. Conversion "n" in process.
NOTE: (1) See Figures 2 and 3 for constraints on data valid from conversion "n-1".			

Table I. Control Line Functions for "Read" and "Convert".

Astfel aplicând simultan un 0 logic pe liniile CS# şi R/C# şi menţinând cel puţin 40 ns linia R/C# în zero logic realizăm eşantionarea şi conversia semnalului analogic. Semnalul BUSY va sta în zero logic din momentul iniţierii conversiei şi până la momentul la care aceasta s-a încheiat. Convertorul îl va aduce automat pe 1 logic, moment la care valoarea numerică corespunzătoare semnalului de intrare va fi validă pe ieşirea convertorului.

Linia BYTE va permuta biţii inferiori şi superiori ai rezultatului. Astfel dacă BYTE=0, pinii de ieşire ai convertorului de la 6 la 13 vor scoate semi-byte-ul mai semnificativ, pentru ca dacă BYTE = 1, aceiaşi pini să reţină cei mai puţini 8 biţi ai rezultatului conversiei.

Structura schemei electrice este gândită pentru procesorul AT89C2051, în urma analizei resurselor pe care acesta le posedă, precum şi a necesităţilor datorate convertorului analog-digital ADS7821 (figura 7.15).

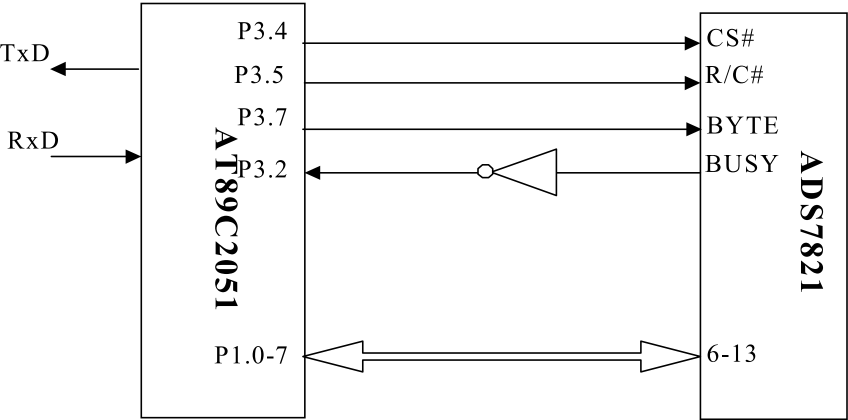


Figura 7. 15 Schema de conectare

Strategia urmată va consta în:

1. declanşarea conversiei prin resetarea liniilor CS# şi R/C#
2. trecerea după mai mult de 40 ns a semnalului R/C#în 1 logic
3. citirea rezultatului atunci când linia BUSY# va trece în 1 logic.

Pentru implementarea achiziției vom putea folosi atât unul cât şi celălalt dintre controller-e.

Să analizăm avantajele şi dezavantajele acestor implementări.

Legând liniile CS#, R/C# şi BYTE la portul 3 al microcontroller-ului nu afectăm funcționalitatea acestuia căci cele două canale temporizatoare le vom folosi, unul pentru generarea ratei de eşantionare a semnalului analogic, iar celălalt drept generator al frecvenței de ceas pe linia serială de date a UART.

Liniile portului P1 sunt direct conectate la liniile 6 la 13 ale convertorului.

Eşalonarea în timp a evoluției semnalelor este dată în figura 7.16. Semnalul byte permută pe acelaşi grup de 8 linii de ieşire byte-ul mai semnificativ cu cel mai puțin semnificativ, funcție de nivelul logic 1, respectiv 0 logic.

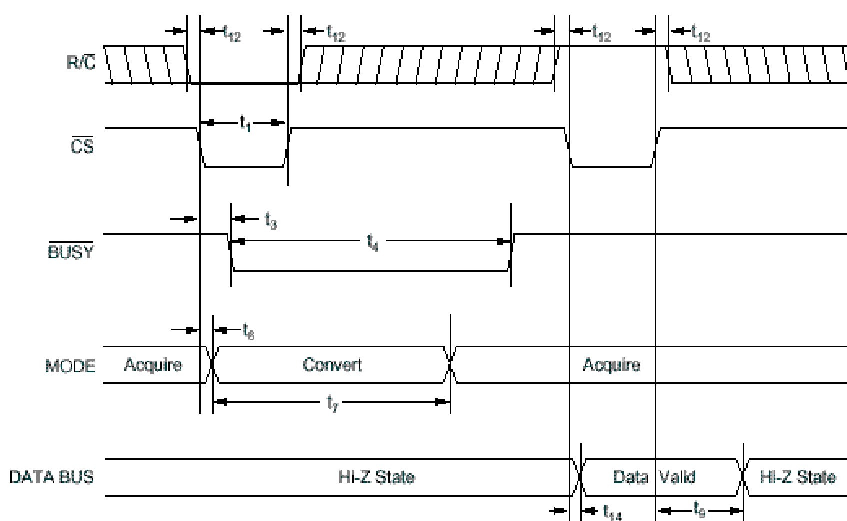


Figura 7. 16 Eşalonarea în timp a semnalelor de comandă pentru interfața ADS7821

Pentru achiziția datelor va trebui să alocăm resursele disponibile ale microcontroller-ului astfel:

1. rutina de servire a întreruperilor corespunzătoare semnalului BUSY, astfel ca pe frontul pozitiv al acestuia să se declanşeze rutina de servire. (vezi declanşarea se face pe frontul descrescător, iar starea de interes la achiziție este starea corespunzătoare frontului crescător, deci am introdus un inversor).
2. rutina de start al conversiei A/D –rutina de servire corespunzătoare întreruperilor timer overflow de la canalul 0.
3. vom scrie rutina de servire a întreruperilor corespunzătoare UART

4. vom proiecta rutina de inițializare
5. vom scrie programul principal de funcționare ce cuprinde rutina de analiză a informațiilor vehiculate prin UART

Programul ce implementează această aplicație a considerat următorul protocol:

- Caracterul „A” reprezintă comanda de achiziție necondiționată a datei
- Caracterul „C” reprezintă comanda de achiziție continuă cu o rata fixă de eşantionare, aprioric stabilită prin constantele de timp CT_L şi CT_H a datelor
- Caracterul „D” reprezintă comanda de transfer a datei achiziționate via interfața UART.

Dăm în continuare programul:

```
CT_L        EQU   00H
CT_H        EQU   80H
CT_SMOD           EQU   21H   ;
CT_ST       EQU   55H   ;
CT_SSCON          EQU   5CH   ;
;Valori pentru constanta de timp necesara la declansarea conversiei semnalului analogic
            DSEG
            BSEG
RX          BIT   10H   ;Specifica receptia unui caracter
TX          BIT   11H   ;Specifica necesitatea transferului DATA_LSB
DE          BIT   12H   ;Specifica existenta unei date achizitionate
DATA_UART   DATA  30H   ;Locatie data receptionata pe UART
DATA_LSB    DATA  31H   ;Locatie LSB data achizitionata
DATA_MSB    DATA  32H   ;Locatie MSB data achizitionata
            CSEG
;ISR_EXT0 Rutina de servire a intreruperilor corespunzatoare achizitiei de la CAD
            ORG   0H
            JMP   MAIN
            JMP   ISR_EXT0
            ORG   0BH
            JMP   ISR_T0
            ORG   23H
            JMP   ISR_SI
ISR_EXT0:
            PUSH  PSW         ;Salveaza starea UC
            SETB  PSW.3
;Seteaza bancul 1 de regsitrii generali, ca registrii "tinta"
            MOV   DATA_MSB,P1       ;Salveaza partea mai semnificativa a datei
```

```
                SETB  P3.7          ;Comanda switch-area bytes: LSB <-
>MSB
                MOV   DATA_LSB,P1 ;Salveaza partea mai putin semnificativa a
datei
                SETB  DE
                POP   PSW           ;Reface starea UC
                RETI                ;Iese din ISR
;ISR_T0 Rutina de servire a intreruperilor corespunzatoare frecventei de
esantionare
ISR_T0:
                PUSH  ACC           ;Salveaza acumulatorul
MOV    A,P3
                ANL   A,#0E7H
                MOV   P3,A          ;Genereaza simultan CS#=0 si R/C#=0
                POP   ACC           ;Reface acumulatorul
                SETB  P3.5          ;Pregatim citirea datei coresp.valorii
analogice
                MOV   TL0,#CT_L
                MOV   TH0,#CT_H     ;Scriu valorile de temporizare
                RETI                ;Iese din ISR
;ISR_SI Rutina de servire a intreruperilor corespunzatoare UART
ISR_SI:
                JB    RI,RECEPTIE   ;Testare intrerupere la receptie
;NU! Intrerupere la transmisie
TRANSMISIE:
                CLR   TI            ;Sterg flag pentrua reanclasa intreruperea
                JB    TX,SCRIE      ;Test daca trebuie sa transmit si LSB
;NU!
                RETI                ;Iesire din ISR
SCRIE:
                MOV   SBUF,DATA_LSB       ;Scriu LSB
                CLR   TX            ;Sterg flag atentionare transfer via UART
a LSB
                RETI                ;Iesire din ISR
RECEPTIE:
                CLR   RI            ;Sterg flag receptie data pe UART
                MOV   DATA_UART,SBUF     ;Citesc data receptionata pe linia
seriala
                SETB  RX            ;Semnalez receptia si citirea datei
                RETI
;Programul principal de functionare a sistemului
MAIN:
                CALL  INIT
LOOP:
```

```
                MOV   A,DATA_UART
                CJNE  A,'A',TEST1
;Comanda SOC ("Start Of Conversion")
                MOV   A,P3
                ANL   A,#0E7H
                MOV   P3,A          ;Genereaza simultan CS#=0 si R/C#=0   ;
                SETB  P3.5
TEST1:
                CJNE  A,'C',TEST2
                MOV   A,P3
                ANL   A,#0E7H
                MOV   P3,A          ;Genereaza simultan CS#=0 si R/C#=0   ;
                POP   ACC           ;Reface acumulatorul
                SETB  P3.5          ;Pregatim citirea datei coresp.valorii
analogice
                MOV   TL0,#CT_L
                MOV   TH0,#CT_H     ;Scriu valorile de temporizare
                SETB  TR0           ;Validez impulsurile catre temporizator -
canalul 0-
                SETB  IE.1          ;Validarea intreruperilor "timer 0
overflow"
TEST2:
                CJNE  A,'D',FINAL
                SETB  DE
FINAL:
                JNB   DE,LOOP
INIT_TRS:
                JB    TI,INIT_TRS
                MOV   SBUF,DATA_MSB
                SETB  TX
                CLR   DE
                JMP   LOOP
INIT:
                MOV   R0,7FH
LP:
                MOV   @R0,#0H
                DJNZ  R0,LP
;Reseteaza memoria interna a controller-ului
                MOV   SP,#60H
                MOV   TMOD,#CT_SMOD
                MOV   TCON,#CT_ST
                MOV   SCON,#CT_SSCON
                MOV   IE,#91H
                MOV   P3,#0FFH
```

RET

END

Locatarea variabilelor şi a registrelor utilizate de către aplicaţie este dată pe CD.

7.1.3 Convertoare A/D complexe

Sistemele de achiziţie ce cuprind convertoare analog-digitale complexe sunt utilizate în aplicaţii cum ar fi achiziţia semnalelor în tensometrie, în domeniul medical, în domeniile ce necesită frecvenţe de eşantionare relativ reduse (de maximum câţiva KHz) şi rezoluţii ridicate (de la 18 la 24 de biţi).

Dintre realizările tehnologice actuale ne vom opri asupra circuitului ADS1211, cap de serie al convertoarelor de înaltă rezoluţie al firmei Burr-Brown/Texas Instruments.

Structura acestuia este prezentată în figura 7.17.

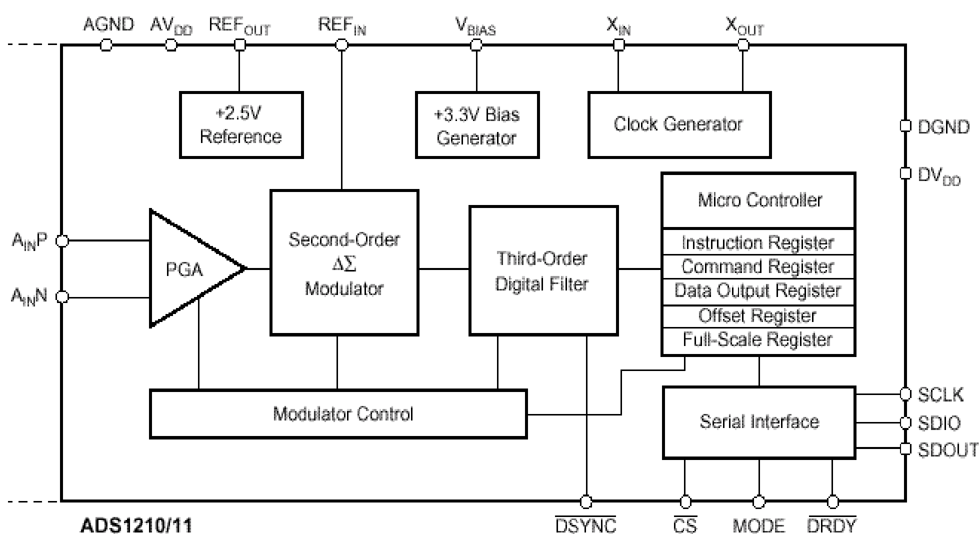


Figura 7. 17 Structura convertorului sigma/delta ADS1210

Circuitul ADS1210/1211 este un convertor analog-digital cu un domeniul de intrare deosebit de larg.

Oferă o rezoluţie de 24 de biţi, cu facilitatea de autocalibrare şi de multiplexare internă diferenţială pentru 4 canale distincte în cazul ADS1211.

Circuitul posedă un amplificator cu amplificare programabilă digital (PGA), un modulator sigma-delta de ordinul 2, un filtru digital programabil care

include un microcontroller ce dispune de regiştrii de instrucţiuni, comenzi, precum şi de regiştrii de date, offset şi calibrare la capăt de scală.

Referinţa de tensiune este internă şi setabilă digital cu valori de până la 2,5V. Funcţionarea detaliată a convertorului este prezentată în foaia sa de catalog. (www.burr-brown.com) unde sunt detaliate şi modurile sale de funcţionare. Două sunt tipice: master mode (MODE=1) -caz în care convertorul are iniţiativa transferului datelor achiziţionate imediat după faza de iniţializare şi slave-mode, (MODE=0)caz în care iniţiativa şi frecevnţa de ceas sunt oferite de către controller-ul magistralei de conexiune.

În figura 7.18 dăm diagramele de timp pentru cazul în care circuitul este conectat ca element slave (pinul MODE este plasat pe 0 logic) pe bus.

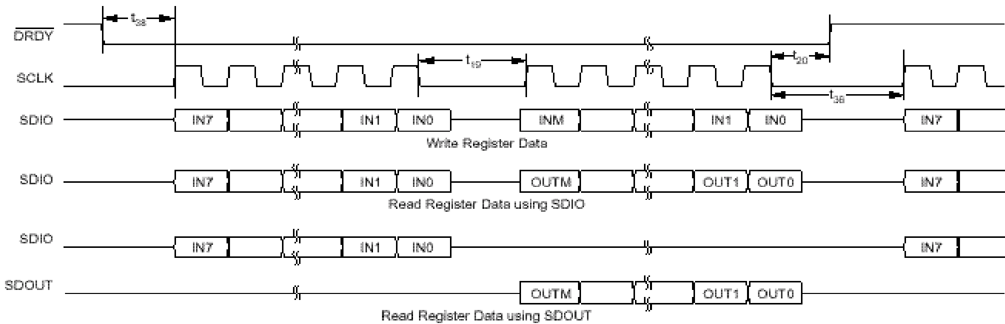


Figura 7. 18 Diagramele de timp corespunzătoare interfeţei seriale

Schema de conectare aleasă permite realizarea tuturor modurilor de funcţionare pentru convertor, respectiv atât a modului "slave" cât şi a modului "master". De asemenea prin controlul liniei CS# este posibilă inserarea mai multor convertoare pe acelaşi bus şi adresarea acestora. În figura 7.19 este prezentată această schemă.

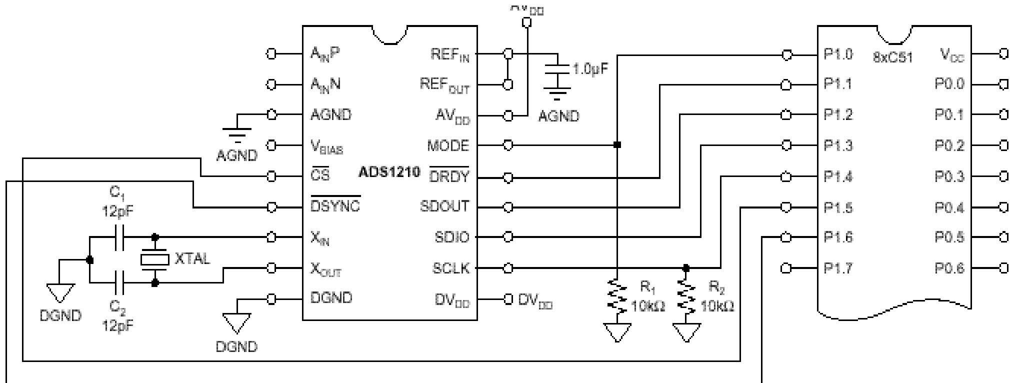


Figura 7. 19 Schema electrică simplificată a interfeţei CAD-ADS1210 microcontroller I8051

Pentru a putea scrie programele de inițializare şi cele de achiziție este necesar să cunoaştem structura regiştrilor interni ai convertorului.

Registrul de instrucțiuni este cel prin intermediul căruia se controlează transferul informațiilor şi are structura:

INSTR

R/W#	MB1	MB0	0	A3	A2	A1	A0

Bitul R/W# specifică tipul operației: de citire/scriere respectiv 1/0 logic.
Biții MB1 şi MB2 codifică numărul de octeți de comandă pe care circuitul îi va recepționa, via interfața serială: combinația 00 corespunde unui octet, 01 la doi octeți, 10 la 3 octeți şi 11 la 4 octeți transmişi succesiv.
Biții A3,A2,A1,A0 adresează regiştrii interni ai convertorului, conform tabelului alăturat.

A3	A2	A1	A0	REGISTER BYTE
0	0	0	0	Data Output Register Byte 2 (MSB)
0	0	0	1	Data Output Register Byte 1
0	0	1	0	Data Output Register Byte 0 (LSB)
0	1	0	0	Command Register Byte 3 (MSB)
0	1	0	1	Command Register Byte 2
0	1	1	0	Command Register Byte 1
0	1	1	1	Command Register Byte 0 (LSB)
1	0	0	0	Offset Cal Register Byte 2 (MSB)
1	0	0	1	Offset Cal Register Byte 1
1	0	1	0	Offset Cal Register Byte 0 (LSB)
1	1	0	0	Full-Scale Cal Register Byte 2 (MSB)
1	1	0	1	Full-Scale Cal Register Byte 1
1	1	1	0	Full-Scale Cal Register Byte 0 (LSB)
Note: MSB = Most Significant Byte, LSB = Least Significant Byte				

Circuitul este astfel conceput încât după fiecare sesiune de scriere adresează implicit, din nou, regsitrul de instrucțiuni.

Odată declaşată, spre exemplu achiziția datelor, acestea vor fi emise autonom, funcție de setările stabilite.

Registrele de comenzi permit setarea parametrilor de achiziție pentru convertorul analog-digital. Între aceştia amintim: rata de eşantionare în concordanță cu frecvența de decimare şi câştigul amplificatorului de pe intrare obținute pe convertor, canalul selectat, formatul datelor, sursa pentru tensiunea de referință şi modul de achiziție: unipolar sau bipolar, ordinea de emisie a datelor începând cu LSB (Last Signifiant Bit) sau cu MSB (Most Signifiant Bit). Setarea modurilor de funcționarese face conform structurii de biți de comandă următori.

Byte 3 (MSB)

BIAS	REF	DF	U/B#	BD	MSB	SDL	DRDY/ DSYNC#
O = Off	1 = On	1 =comp 2	0= Bipolar	0= MSB	0=MSB	0=SDIO	0

Valorile trecute pe a doua linie sunt cele implicite.

- Pentru valoarea 0 a bitului BIAS, tensiunea de referință este de 2,5V, iar pentru valoarea 1 ea este 1,33*2,5V adică aproximativ 3,3V.
- Un 1 logic pe bitul REF asigură utilizarea sursei interne de referință, iar 0 aduce în starea de mare impedanță această sursă, permitând utilizarea unei surse de referință externe.
- DF exprimă formatul datelor, 0 implică formatul datelor în complement față de doi +FSR=7FFFFFH, 0=000000H, iar -FSR=800000H (FSR este abrevierea scalei complete "Full Scale Range").
- U/B# exprimă când este 0 achiziția bipolară a datelor, iar când este 1 exprimă achiziția unipolară a acestora.
- BD exprimă ordinea de transfer a bytes, setând bitul ordinea este crescătoare de la LSB către MSB şi invers dacă-l resetăm.
- MSB exprimă cine este primul bit emis în cadrul fiecărui byte, 1 implică transferul întâi al bitului cel mai nesemnificativ, iar 0 implică transferul întâi al bitului cel mai semnificativ.
- SDL comandă utilizarea a unei linii bidirecționale seriale (SDIO) dacă este resetat, respectiv a două linii seriale dacă este setat (SDIO pentru comenzi şi SOUOT pentru date)
- DRDY# exprimă resetat prezența datelor (Data Ready), iar setat exprimă date invalide.

Byte 2

MD2	MD1	MD0	G2	G1	G0	CH1	CH0

Acest byte setează, modurile de funcționare, codate binar de la 0 la 7, valorile câştigului şi selectează canalul de intrare pentru achiziția de semnal.

Modurile de funcționare codate binar sunt în ordine, începând de la 000H următoarele:

1. Normal,
2. "Self calibration" calibrare automată,
3. Calibrare de offset,
4. Calibrare la capăt de scală,
5. "Pseudo-calibrare",
6. Calibrare în fundal "background calibration",
7. "Sleep" şi
8. Rezervat -neutilizat.

Byte 1

SF2	**SF1**	**SF0**	**DR12**	**DR11**	**DR10**	**DR9**	**DR8**

SF2, SF1, SF0 selectează una dintre valorile modului turbo de funcţioare a circuitului, iar valorile DR12 la DR0 setează frecvenţa de decimare pentru filtrul digital al convertorului.

Byte 0

DR7	**DR6**	**DR5**	**DR4**	**DR3**	**DR2**	**DR1**	**DR0**

În modul de conectare ce utilizează liniile SDIO şi SDOUT convertorul poate fi controlat dinamic, respectiv poate lucra în modul de citire continuă, admitând în acelaşi timp comenzi prin intermediul liniei SDIO şi generând date via linia SDOUT. (vezi documentaţia firmei Burr - Brown, memorată pe CD)

Implementarea unui ciclu descriere a comenzilor în modul slave presupune îndeplinirea următorilor paşi:

1. Aşteaptă ca ADS1211 să plaseze DRDY# pe zero logic
2. Aşteaptă ca CS# să treacă la rândul său în zero logic
3. Generează 8 cicluri de ceas pentru a trece în instruction register data de configurare via SDIO
4. Generează **n** ciclii de ceas de transmisie pentru înscrierea regiştrilor ţintă ai comenzilor (vezi documentaţia la ADS1210 ads1210.pdf de la Burr-Brown)
5. Ads1210 pune pe 1 logic DRDY

Repetă paşii 3 la 5 până la transmiterea tuturor cuvintelor de comandă către ADS1210, după care intră în ciclul/ciclurile de citire a datelor, ceea ce presupune:

1. Aşteaptă ca ADS1211 să plaseze DRDY# pe zero logic
2. Aşteaptă ca CS# să treacă la rândul său în zero logic
3. Dacă este în modul citire continuă (Continous Read), trece la pasul 5
4. Microcontroller-ul generează 8 ciclii de ceas şi transferă cuvântul de control în registrul de instrucţiuni via SDIO
5. Dacă utilizăm SDIO drept linie bidirecţională vom urmării evoluţia acesteia în continuare, dacă nu, atunci linia SDIO trece în starea de mare impedanţă
6. Microcontroller-ul emite **n** ciclii de ceas (**n** multiplu de 8) şi transmite/recepţionează informaţia din registrul specificat prin instrucţiunea de la punctul 4. Recepţia acestor biţi se face pe linia SDOUT sau SDIO după caz.
7. SDOUT sau SDIO intră în starea de mare impedanţă, după ce a transmis informaţia comandată

8. Testare dacă mai sunt de transmis alte comenzi sau cereri de date. Dacă **da** se reia ciclul de la punctul 2. CS# rămâne inactiv 10,5 ciclii de ceas.

Programul de mai jos prezintă câteva rutine, ca de transfer a unui cuvânt de comandă, precum şi cea de citire a unei date formată din trei octeți. Programarea microcontroller-ului va fi specifică pentru fiecare aplicație în parte, ea trebuind să realizeze adaptarea parametrilor convertorului în raport cu rezoluția şi rata de eşantionare impuse de aplicație.

Programul de funcționare permite:

1. Sincronizarea transferului comenzilor către CAD;
2. Generarea soft şi emiterea semnalului de ceas (SCLK), către CAD
3. Transferul informațiilor de inițializare şi a comenzilor către CAD
4. Recepția datelor corespunzătoare semnalului analogic convertit

Iată acest program:

```
        ORG     28H
PROG:   DB      01100100B       ;Cuvânt de instrucțiuni
        DB      01000010B       ;Cuvânt de C-dă nr. 3
        DB      00000000B       ;Cuvânt de C-dă nr. 2
        DB      00000000B       ;Cuvânt de C-dă nr. 1
        DB      00000000B       ;Cuvânt de C-dă nr. 0
;Cuvântul de instrucțiuni asigură scrierea cuvintelor de comandă 3, 2, 1 şi 0
        DB      11000000
;Cuvântul asigură citirea celor 3 regiştrii corespunzători valorii semnalului
analogic convertit
BUFFER:         DS      03H
CAD:            JB      P1.0,CAD        ;Testează starea CAD, respectiv DRDY#=0
        MOV     R1,#04H
        MOV     R0,#PROG
;Pregăteşte transmiterea a 5 octeți către CAD, ce sunt: cuvântul de instrucțiuni şi
;cele 4 cuvinte de comandă. Cuvintele sunt memorate începând de la adresa PROG
LOOP_1:         MOV     A,@R0
        CALL    SERIAL_OUT
        INC     R0
        DJNZ    R1,LOOP_1
CAD1:   JB      P1.0,CAD1       ;Testează starea CAD, respectiv DRDY#=0
        MOV     A,@R0
        CALL    SERIAL_OUT
LOOP_2:         JB      P1.0,LOOP_2
;Aşteaptă conversia datelor de către CAD, semnalul DRDY# va fi resetat
        MOV     R1,#03H
        MOV     R0,#BUFFER
;Pregăteşte citirea celor 3 bytes de date, corespunzători semnalului analogic
LOOP_3:
```

```
        CALL  SERIAL_IN
        MOV   @R0,A
        INC   R0
        DJNZ  R1,LOOP_3
;Încheie un ciclu complet Inițializare comandă şi achiziție data convertită
SERIAL_OUT: CLR    P1.3
        MOV   R7,#08H
        RL    A
        RL    A
LOOP_OUT:
        JB    ACC.2,SET_OUT
        CLR   P1.2
        JMP   CLOCK_OUT
SET_OUT:    SETB   P1.2
CLOCK_OUT:  SETB   P1.3
        CRL   P1.3              ;Generează semnalul SCLK
        RL    A
        DJNZ  R7,LOOP_OUT           ;Transferă un octet către CAD
        RET
SERIAL_IN:  CLR    P1.3
        MOV   R7,#08H
LOOP_IN:
        JB    P1.1,SET_IN
        CLR   ACC.0
        JMP   CLOCK_IN
SET_IN:     SETB   ACC.0
CLOCK_IN:   SETB   P1.3
        CRL   P1.3
        RL    A
        DJNZ  R7,LOOP_IN            ;Receptează un octet de la CAD
        RET
```

7.1.4 Convertor D/A controlabil serial

Aplicația pe care o vom descrie în continuare se referă la comanda unui convertor digital - analog cu intrare serială, cum ar fi cele utilizate în cadrul aparaturii audio de înaltă performanță. Vom exemplifica aceasta prin circuitul PCM56 convertor digital-analog de 16 biți. Structura convertorului este dată în figura 7.20. Câteva dintre caracteristicile convertorului sunt: domeniul dinamic de peste 96 dB FSR, nu necesită componente externe, 16 biți rezoluție, eroare liniară sub 0,001%, timp de stabilire 1,5 ms, operează cu tensiuni între ±5V şi ±12V. LE

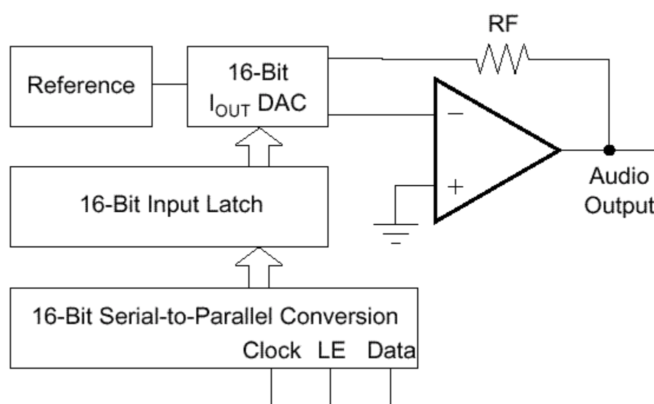


Figura 7. 20 Structura convertorului PCM56 (Fabricat de Burr-Brown)

"Latch Enable", lăcătuieşte informația serial receptată şi o aplică convertorului la momentul validării lăcătuirii.

Data este intrarea de "Date", ce sunt transmise în format complement față de 2, sincron cu semnalul de ceas "Clock". Amănunte privind fronturile şi sincronizarea datelor şi timming-ul corespunzător transferului unei date sunt prezentate în figura 7.21. Se constată că datele sunt transmise începând cu cel mai semnificativ octet, frontul negativ realizează deplasarea acestora la dreapta bit de

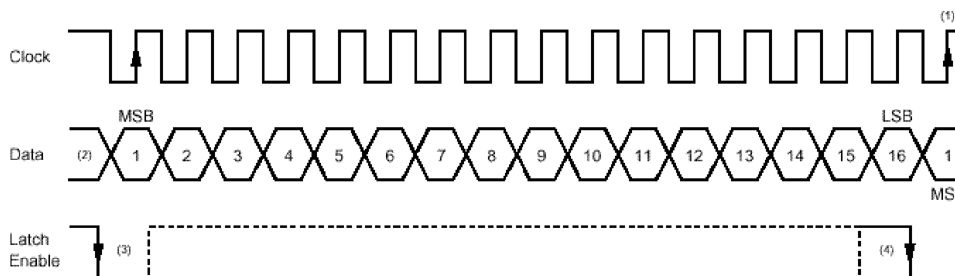


Figura 7. 21 Diagramele de timp ce ilustrează comanda convertorului PCM65

bit. Frontul pozitiv reprezintă frontul pe care data este memorată în registrul de deplasare aflat pe intrarea convertorului digital-analog PCM56. Intrarea LE trebuie să stea pe 1 logic cel puțin un ciclu de ceas, ea plasându-se pe durata transferului serial al datelor pe 0 logic. La sfârşitul transferului ea va fi trecută în 1 logic pentru cel puțin 1,5 perioade de ceas.

Utilizând microcontroller-ul AT89C2051 vom conecta liniile P1.7 la linia Clock, P1.6 la linia Date, iar P1.5 la linia LE. Dăm mai jos rutina care realizează comanda convertorului.

```
CT_TMOD             EQU   21H
;Setare mod functionare canale temporizatoare:
;        canalul 0 utilizat pentru rata de eşantionare comandă programat în modul 1
;        canalul 1 utilizat pentru rata de transfer serial programat în modul  2
CT_TCON             EQU   50H
;Setez flag-urile de dezinhibare a numărării pentru canalele 0 & 1
CT_SCON             EQU   90H
;Setez modul 1 de transfer serial UART 8 biți pilotat cu rata variabilă de către
;canalul 1 temporizator
CT_LOW0             EQU   00H
CT_HIGH0     EQU    00H
CT_LOW1             EQU   0FDH
;Constanta de timp corespunzătoare ratei de 9600 Baud la fceas=11,059MHz
LEN_BUFFER          EQU   20H            ;Lungime prestabilită a blocului
;de date de transferat.   CONTOR_B este mai mic sau egal cu LEN_BUFFER
             BSEG
REC          BIT    0H
;Flag-ul setat exprimă recepția completă a unui caracter via UART
RANG_DATA    BIT    1H
;Flag ce exprima byte-ul care este transferat către CDA
             DSEG
DATA_CDA:    DS     LEN_BUFFER ;Buffer bloc date
DATA_LOW     DATA   00H     ;LSB corespunzător datei de convertit D/A
DATA_HIGH    DATA   00H     ;MSB corespunzător datei de convertit D/A
;Locații corespunzătoare datei transferate către CDA în modul transfer cuvânt
DATA_SI      DATA   00H     ;Buffer de recepție a datelor (comenzi) via UART
CONTOR_B     DATA   00H     ;Mărime bloc de date de transferat către CDA
POINTER_D    SET    DATA_CDA
             CSEG
             ORG    000H
BOOT:
             JMP    MAIN
             ORG    0BH
ISRT0:
             JMP    ISR_T0
             ORG    23H
ISRSI:
             JMP    ISR_SI
;Rutina corespunde transferului unui cuvânt către CDA
ISR_T0:      PUSH   PSW
             PUSH   ACC              ;Salvez starea microcontroller-ului
```

```
                SETB  PSW.3          ;Utilizez bancul 1 de regiştrii generali
                CLR   P1.5
;Invalidez LE, => Posibilitate de scriere in buffer-ul de serializare al CDA
                MOV   R7,08H
;Încarc contor al biților de serializat corespunzător MSB
                MOV   A,DATA_HIGH          ;Încarc MSB
LOOP_CDA1:      RL    A              ;Rotesc informația din Acumulator
                CLR   P1.7           ;Reset semnal Clock către CDA
                JB    ACC.0,SET_BIT_CDA
                CLR   P1.6           ;Reset DATA către CDA
CLK_CDA:        SETB  P1.7           ;Set semnal Clock către CDA
                DJNZ  R7,LOOP_CDA1
                JMP   CONT
SET_BIT_CDA:          SETB  P1.6           ;Set DATA către CDA
                JMP   CLK_CDA  ;Am transmis MSB către CDA, urmează LSB
CONT:           MOV   R7,08H
                MOV   A,DATA_LOW
LOOP_CDA2:      RL    A
                CLR   P1.7           ;Reset semnal Clock către CDA
                JB    ACC.0,SET_BIT_CDA1
                CLR   P1.6           ;Reset DATA către CDA
CLK_CDA1:       SETB  P1.7           ;Set semnal Clock către CDA
                DJNZ  R7,LOOP_CDA1
                SETB  P1.5
;Lăcătuiesc informația transferată serial în buffer-ul de ieşire al CDA
                CLR   P1.5
;Revalidez accesul la buffer-ul de intrare (serial) al CDA
                POP   ACC
POP     PSW                  ;Refac starea UC dinaintea intreruperii de timp
RETI
SET_BIT_CDA1:
                SETB  P1.6           ;Set DATA catre CDA
                JMP   CLK_CDA1
ISR_SI:
                JB    RI,RECEPTIE
TRANSMITE:
                CLR   TI
                RETI
RECEPTIE:
                MOV   DATA_SI,SBUF
                CLR   RI
                SETB  REC
                RETI
MAIN:
```

```
                MOV   R0,#7FH
LP:
                MOV   @R0,#00H
                DJNZ  R0,LP
;Scriu 0 in toate locatiile corespunzatoare memoriei interne a controller-ului
                MOV   SP,70H
;Setez adresa pentru stiva sistemului
                CALL  INIT
;Realizez initializarile canalelor temporizatoare si a interfetei seriale
                MOV   IE,#92H
;Validez întreruperile corespunzătoare canalului 0 temporizator şi UART
LOOP:
;Se va particulariza programul funcție de protocolul dorit şi funcție de
;particularitățile sistemului
                JNB   REC,LOOP
;Testare caracter recepționat şi comanda corespunzatoare codului asignat
; ................
                CLR   REC
;Reanclaşare analiză a unui eventual nou caracter recepționat via UART
                JMP   LOOP
INIT:
                MOV   TMOD,CT_TMOD
                MOV   TCON,CT_TCON
                MOV   TL0,#CT_LOW0
                MOV   TH0,#CT_HIGH0
                MOV   TL1,#CT_LOW1
                SETB  0D7H
;Dublez rata de transfer serial a informațiilor =>PCON.7=SCON=1
                RET
END
```

7.1.5 Programarea unui sistem de dozare gravimetrică dotat cu microcontroller

Aplicația pe care o prezentăm reprezintă un cântar electronic necesar dozarii materiilor prime pulverulente, bazat pe utilizarea unei balanțe electronice ce prezintă o doză tensometrică compensată complet cu ajutorul căreia se determină greutatea ansamblului cântar şi pulbere (figura 7.22).

Sistemul electronic de dozare a materiilor prime pulverulente este format din două subsisteme principale:

1. Subsistemul mecanic, sub forma unui cântar diferențial de precizie

2. Subsistemul electronic de măsurare a tensiunii apărute în reazemul către sistemul de referință fix format din cadrul cântarului.

Principiul de funcționare al cântarului dozator este acela de transformare a greutății corespunzătoare materiilor prime pulverulente în semnal electric prin

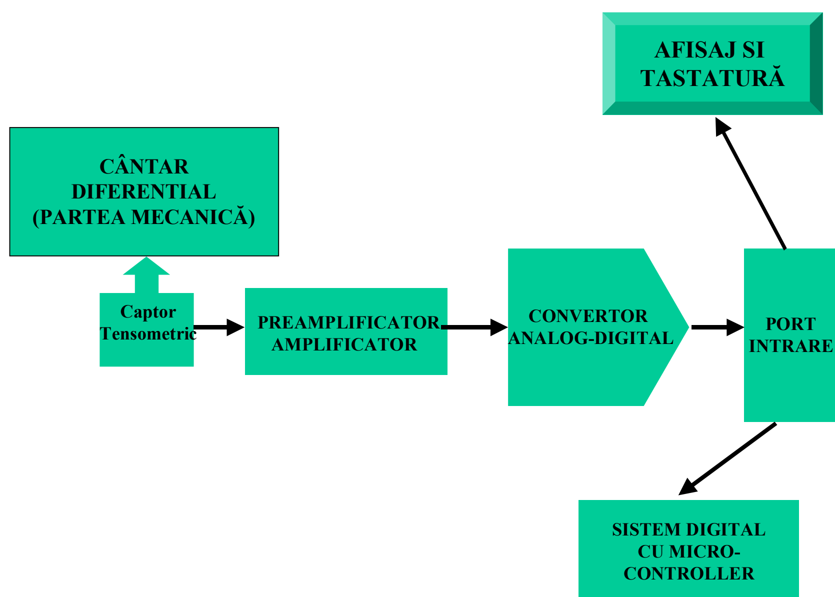


Figura 7. 22 Structura dozatorului gravimetric

intermediul unui captor de forță (Philips PR6211)**.** Acesta oferă un semnal de tensiune în diagonala unei punți tensometrice rezistive care este alimentată de la o sursă de tensiune de referință. Sensibilitatea senzorului este de 2mV/V, rezultând la o încărcare egală cu cea maxim admisă, şi cu o tensiune de alimentare a punții de 5V rezultă o diferență maximă de tensiune de 10mV pe diagonala punții de măsurare.

Prelucrarea semnalului analogic de la captorul tensometric este realizată în două trepte de amplificare şi filtrare. Pentru semnalul amplificat, - domeniul de variație este 0-10V – el fiind aplicat unui convertor analog-digital ADS7805 (analog convertorului ADS7821), de 16 biți ce–l converteşte în semnal digital.

Sistemul digital este coordonat de către un microcontroller de 8 biți Philips PCB80C552 (analog unuia Intel 8051), ce dispune de 40 de linii digitale de intrare/ieşire, la care se adaugă 3 canale numărătoare/temporizatoare, precum şi un convertor analog/digital de 10 biți intern precum şi două ieşiri PWM (Pulse Width Modulation). Microcontroller-ul posedă două interfețe seriale, una standard UART

(Universal Asyncronus Receiver Transmiter), şi una I^2C care asigură o rată de transfer mai bună de 4Mbiţi/secundă. Memoria internă a controller-ului este de 256 de bytes la care se adaugă regiştrii de funcţii speciale (Special Function Registers).

Sistemul dispune de două afişoare, unul cu LED-uri cu şapte segmente, necesar afişării greutăţii măsurate, iar al doilea cu LCD-uri 2x16 caractere necesar pentru ghidarea prin intermediul mesajelor a operatorului, precum şi o tastatură cu 12 taste multifuncţionale.

Memoria sistemului este de 64Kbytes, dintre care 32Kbytes EPROM şi 32Kbytes SRAM.

Subsistemul mecanic asigură transmiterea greutăţii materiilor prime pulverulente către captorul tensometric prin demultiplicarea efortului pe acesta. Factorul de demultiplicare al cântarului are expresia: $\kappa = a\frac{1}{2(a+b)+\delta}$, unde: a,b şi δ sunt dimensiuni ale braţelor cântarului diferenţial. Pentru dimensiunile de a=78 mm, b=300 mm şi δ=150 mm, factorul **k** va fi: k=0,08,

Subsistemul electronic de măsurare include captorul de forţă ce traduce reacţiunea normală aplicată asupra sa în semnal electric de tensiune.

Iată principalele caracteristici ale elementelor subsistemului:

Captorul de forţă: clasă de precizie: D1, respectiv 0,05% eroare integrală de măsurare, (aceasta include toate erorile cea de zero, de neliniaritate, cea corespunzătoare variaţiei temperaturii, etc). Rezistenţa de intrare a captorului 1200 Ω, rezistenţa de ieşire 1200 Ω, rezistenţa de izolaţie a mărcilor captorului în raport cu masa acestuia mai bună 5000MΩ. Tensiunea de alimentare a punţii tensometrice este de 5V cc, obţinuţi de la o sursă de tensiune REF02 (Burr Brown) având o stabilitate excelentă în domeniul de temperaturi de la - 10 la +85°C. Preamplificatorul de semnal este implementat cu ajutorul unui circuit INA114(Burr Brown), amplificator de măsurare integrat ce nu necesită decât o singură rezistenţă externă de setare a amplificării. Dintre caracteristicile sale amintim: domeniu de tensiuni de alimentare: ±2,25V la ±18V, tensiune maximă de offset 50μV, drift maxim 0,25μV/°C, CMRR mai bun de 115dB, curent maxim de polarizare pe intrări 2nA, domeniu temperaturilor de funcţionare între -40 şi +85°C, este protejat la scurtcircuit pe ieşire, şi la supratensiuni pe intrare. Circuitului i s-a setat o amplificare de 1000.

Amplificatorul de semnal: este implementat cu acelaşi tip de amplificator ca şi preamplificatorul, amplificarea acestuia fiind de 1,1316.

Convertorul analog-digital ADS7805, cu următoarele caracteristici: 16 biţi rezoluţie, eroarea maximă integrală 2-3 LSB (Last Signifiant Bit), fără coduri lipsă, necesită doar tensiunea de 5V pe alimentare, domeniul de intrare ±10V, codarea în format complement faţă de 2, rata de eşantionare maximă 100 kHz, ieşirea

convertorului pe 16 biți se poate cupla direct la magistrala de date a sistemului digital, domeniul temperaturilor de funcționare –40 la +85°C.[13]

Sistemul digital: asigură comanda şi controlul de ansamblu, permite implementarea funcțiunilor de măsurare, prelucrare, memorare, dozare şi afişare a greutății materiilor prime pulverulente.

Unitatea centrală PCB80C552, dispune de o memorie internă de 256 bytes, iar sistemul de o memorie externă de 64Kbytes(32K EPROM /32KSRAM). Pentru a se realiza funcțiunile sistemului s-au adăugat porturi digitale de ieşire şi intrare de câte 8 biți fiecare, implementate cu circuitele 74LS374, 74LS574, respectiv 74LS244 şi 74LS574 care au o capabilitate în curent la ieşire de 20-24mA/ieşire. Afişajul cu LCD-uri este implementat cu circuitul LTN211, circuit ce posedă un microcontroller, memorie internă şi un generator de caractere necesar afişării a până la 512 caractere diferite. Capacitatea acestuia este de 2x16 caractere. Afişorul cu LED-uri este clasic, transferul informațiilor binare către acesta realizându-se serial (vezi schema pe CD). S-a prevăzut o placă specială de ieşiri numerice pe relee (4 bucăți cu câte 2 contacte fiecare de câte 8 A), şi 8 intrări digitale izolate galvanic. Tastatura sistemului cuprinde 12 taste scanate soft de către procesor.

Dintre funcțiunile pe care sistemul le implementează amintim:

- Dozarea în fluxul tehnologic a materiilor prime pulverulente
- Filtrarea automată a datelor de măsurare prin medierea unei serii de eşantioane.
- Tararea automată realizată la începutul procesului de dozare, sau la cereera operatorului.

Alte funcțiuni implementate:

- Presetare greutate de dozat
- Presetare tempozare la sfârşitul procesului de dozare (rezoluție 1 secundă, domeniu 0 la 999s)
- Presetare tempozare pentru pornirea amestecătorului (rezoluție 1 secundă, domeniu 0 la 999s)
- Presetare tempozare la sfârşitul procesului de golire a cântarului (rezoluție 1 secundă, domeniu 0 la 999s)
- Determinare TARA
- Afişare presetare greutate de dozat
- Afişare presetări temporizări
- Setare constantă de calibrare
- Afişare constantă de calibrare

---

[13] *ATENȚIE:* se recomandă buffer-area ieşirii convertorului cu buffer-e în tehnologie Shotky pentru a nu permite traversarea convertorului de către glich-uri ce pot apare pe partea analogică de prelucrare a informațiilor. Aceşti paraziți pot afecta funcționarea corectă a sistemului!

- Transfer serial în format binar a greutăţii măsurate în fazele de dozare şi golire a cântarului (formatul datelor: asincron, rata 9600 Bauds, 8 biţi/cuvânt, fără paritate, 2 biţi de stop)

Sistemul permite reglarea anumitor parametrii ai săi, precum: cantitatea dozată, temporizările corespunzătoare diferitelor faze ale procesului de dozare.

Utilizând sistemul de întreruperi al procesorului putem declanşa procesul de conversie la intervale de timp prestabilite pentru convertorul analog-digital de semnal asigurând eşantionarea uniformă a semnalului de greutate.

Programul complet de funcţionare al sistemului este înscris pe CD unde este dată şi schema sistemului utilizat pentru această aplicaţie.

În continuare prezentăm câteva rutine care implementează comanda unui afişor LCD, comanda unui afişor realizat cu afişoare cu câte 7 segmente (5 afişoare), rutina de înmulţire a două cifre de câte 16 biţi fiecare şi explicăm modul de utilizare a canalului "Watch Dog" pentru a evita agăţarea sistemului în cazul apariţiei unor zgomote parazite la nivelul magistralei de sistem.

Scrierea către un modul de afişare LCD (LTN211, conectat la un port paralel de adresă 100H), implică realizarea unor întârzieri necesare procesului de transfer al datelor şi comenzilor. Implementarea acestora s-a realizat soft (întârziere prin execuţia de instrucţiuni), căci pe de-o parte nu este critică durata temporizată, iar pe de altă parte în cadrul aplicaţiei canalele temporizatoare/numărătoare au fost dedicate altor procese ce necesită o precizie mai ridicată. Am utilizat MACRO-uri pentru scrierea acestor programe. În cadrul rutinelor de întârziere am avut grijă să reanclanşăm periodic temporizatorul "Watch Dog", pentru a evita generarea unui RESET al sistemului.

```
;Rutine pentru afisarea pe LCD
EXTERN        WAIT
WAITVL   MACRO
;Macro ce realizează o întârziere de
LOCAL B_0,B_1,B_2                   ;Etichete definite local în cadrul MACRO
         MOV R5,#20H         (12) ;Iniţializează R5 cu valoarea de ciclare 32
B_1:     MOV R6,#0FFH        (12) ;Iniţializează R6 cu valoarea de ciclare 255
B_0:     MOV R7,#0FFH        (12) ;Iniţializează R6 cu valoarea de ciclare 255
B_2:     DJNZ R7,B_2         (24) ;Decrementează contoare pe rând
         DJNZ R6,B_0         (24) ;Decrementează contoare pe rând
         ORL    PCON,#10H(12)
         MOV     T3,#00H     (12)
         DJNZ R5,B_1         (24) ;Decrementează contoare pe rând
ENDM
```

În partea dreaptă a mnemonicelor s-au menţionat numărul de perioade de ceas necesare execuţiei instucţiunilor. Considerând frecvenţa de ceas egală cu 12 MHz, vom obţine temporizarea de: 4,192secunde.

Menţionăm că regiştrii R5, R6 şi R7 sunt utilizaţi drept contoare

```
WAITL    MACRO
```

```
            LOCAL B_0,B_1,B_2
                        MOV R6,#04H
B_0:                    MOV R7,#0FFH
B_2:                     DJNZ R7,B_2
;Anclansare WATCH DOG
                        ORL     PCON,#10H
                        MOV     T3,#00H
                        DJNZ R6,B_0
ENDM                                    ;Temporizarea de: 0,002048secunde.
;Registrul este pointer-ul în buffer-ul de mesaj de scris la afişor LCD
;Registrul DPTR este pointer-ul către portul microsistemului la care se află afişorul
;LCD
BUCLA:
                        MOV A,#0                ;Pointer la buffer-ul de mesaj
                        MOVC A,@A+DPTR          ;Rutina de scriere mesaj la LCD
                        INC DPTR
                        MOV R0,DPL
                        MOV R1,DPH
                        MOV DPL,R3
                        MOV DPH,R4
                        WAIT
                        ORL     PCON,#10H       ; Setare flag corespunzător WATCH DOG
                        MOV     T3,#00H         ;Anclanşare WATCH DOG: scrie constanta
                        MOVX @DPTR,A            ;Scrie caracterul către LCD
                        MOV DPL,R0
                        MOV DPH,R1
                        DJNZ R2,BUCLA
                        MOV  DPL,R3
                        MOV DPH,R4
                        MOV A,#2
                        MOVX @DPTR,A
;Scrie comanda de avans cu un caracter la afişorul LCD
                        WAIT
                        RET
;Rutină de scriere a mesajelor către afişorul LCD
;MESAJ1 şi MESAJ2 adresele de început pentru mesajele de afişat pe primul
;respectiv, pe al doilea rând al afişajului
LCD_PRINT       MACRO       #MESAJ1,#MESAJ2
                        MOV DPTR,#100H                  ;Adresa de scriere la LCD
                        MOV  A,#1                       ;Selecție funcție:
                        MOVX @DPTR,A                    ;DISPLAY / CLEAR LCD
                        WAITL
                        MOV A,#80H                      ;Selecție funcție Cursor plasat pe:
                        MOVX @DPTR,A                    ;RÂNDUL 1
```

```
WAIT
MOV R2,#10H;Încarcă contor caractere ce vor fi afişate pe r-dul 1
MOV DPTR,#101H ;Încarcă adresa de scriere către afişorul LCD
MOV R3,DPL
MOV R4,DPH
MOV DPTR,#MESAJ1 ;Încarcă pointer către mesajul de afişat
MOV R0,DPL
MOV R1,DPH
WAITL
CALL BUCLA ;Rutină de scriere mesaj (lungime mesaj 16 car.)
WAITL
MOV DPTR,#100H
MOV A,#0C0H
MOVX @DPTR,A;      Comandă saltul la rândul 2 al afişorului
WAIT
MOV DPTR,#101H
MOV R3,DPL
MOV R4,DPH
MOV DPTR,#MESAJ2 ;Încarcă pointer către mesajul de afişat
MOV R0,DPL
MOV R1,DPH
MOV R2,#10H            ;Contor de caractere Rând 2
WAITL
CALL BUCLA ;Rutină de scriere mesaj (lungime mesaj 16 car.)
WAITL
 ENDM
```

În continuare prezentăm rutina de înmulțire întreagă a două valori memorate în locațiile OP1 şi OP2 (locații de 16 biți fiecare). În locațiile MREZ1, MREZ2, MREZ3 şi MREZ4 (8 biți fiecare), în ordinea crescătoare a rangului, vom obține rezultatul înmulțirii. Rutina va genera produsul celor doi operanzi, iar flag-ul C va fi afectat la depăşirea buffer-ului rezultat.

Notând cu N1H şi N1L, respectiv cu N2H şi N2L octeții corespunzători celor două numere, acestea pot fi scrise astfel: N1=256*N1H+N1L, iar N2=256*N2H+N2L. (N1L=LOW(1),N2L=LOW(N2), iar N1H=HIGH(N1) N2L=HIGH(N2)).

Înmulțindu-le obținem:

N1*N2=65536*N1H*N2H+256*(N1H*N2L+N2H*N1L)+N1L*N2L. Execuția acestei înmulțiri se va face începând de la dreapta spre stânga, ținându-se cont de eventualul bit de transport ce poate apare la execuția operațiilor.

```
M16:
        MOV     B,OP1L      ;Încarcă în B pe N1L
        MOV     A,OP2L      ;Încarcă în A pe N2L
        MUL     AB          ;Execută N1L*N2L
```

```
                MOV     MREZ1,A   ;Reține LSB rezultat din înmulțirea
        N1L*N2L
                MOV     MREZ2,B   ;Reține MSB rezultat din înm. N1L*N2L
                MOV     A,OP2H
                MOV     B,OP1L
                MUL     AB        ;Execută produsul: N2H*N1L
                ADD     A,MREZ2   ;Adună: LSB(N2H*N1L)+MSB(N1L*N2L)
                MOV     MREZ2,A   ;Salvează rezultatul
                MOV     A,B       ;Transferă în registrul A=MSB(N2H*N1L)
                ADDC    A,MREZ3   ;Adună cu transport în A=MSB(N2H*N1L)
                MOV     MREZ3,A   ;Transferă rezultat în MREZ3
                MOV     A,OP1H
                MOV     B,OP2L
                MUL     AB        ;Execută produsul:N1H*N2L
                ADD     A,MREZ2
;Adună: LSB(N1H*N2L)+LSB(LSB(N2H*N1L)+MSB(N1L*N2L))
                MOV     MREZ2,A   ;Transferă rezulatul în MREZ2
                MOV     A,B       ;Transferă MSB(N1H*N2L) în registrul A
                ADDC    A,MREZ3
                MOV     MREZ3,A
;Reține în MREZ3,MREZ2 şi MREZ1 expresia calculată:
;256*(N1H*N2L+N2H*N1L)+N1L*N2L
;De observat că îmulțirea cu 256 este specificată prin locația în care rezultatele sunt
;salvate, după regula rangul cel mai ridicat este plasat la o adresă mai mare
                MOV     A,OP1H
                MOV     B,OP2H
                MUL     AB        ;Execută produsul N1H*N2H
                ADD     A,MREZ3
                MOV     MREZ3,A   ;Salvez LSB(N1H*N2H)
                MOV     A,B
                ADDC    A,MREZ4
                MOV     MREZ4,A   ;Salvez MSB(N1H*N2H)
                RET
NR_OCT          DATA  13H   ;Index octeți serializați
SERBUF          DATA  21H   ;Locație octet de serializat către afişoare
;Locațiile E3 şi E4 reprezintă biții 3, respectiv 4 ai Acumulatorului
SERPIN1         BIT   0E3H  ;Ieşire clock registru intrare serială/ieşire paralelă
SERPIN0         BIT   0E4H  ;Ieşire date registru de deplasare (MMC4015)
BIT_0           BIT   8     ;Bitul 0 al locației SERBUF (21H)
BIT_1           BIT   9     ;Bitul 1 al locației SERBUF (21H)
BIT_2           BIT   0AH   ;Bitul 2 al locației SERBUF (21H)
BIT_3           BIT   0BH   ;Bitul 3 al locației SERBUF (21H)
BIT_4           BIT   0CH   ;Bitul 4 al locației SERBUF (21H)
BIT_5           BIT   0DH   ;Bitul 5 al locației SERBUF (21H)
```

```
BIT_6           BIT    0EH     ;Bitul 6 al locației SERBUF (21H)
BIT_7           BIT    0FH     ;Bitul 7 al locației SERBUF (21H)
BUF_AF:         DB     01H,23H,45H,67H,89H ;Zonă memorare informații
                DB     11H,11H,11H,11H,11H ;de afişat.
                DB     22H,22H,22H,22H,22H ;Valori de TEST
                DB     33H,33H,33H,33H,33H
; Refresh afişoare
                MOV    SP,#70H           ;Setare adresă de bază stivă
                MOV    NR_OCT,#0         ;Înițializare contor
                MOV    DPTR,#(BUF_AF-1)      ;Inițializare pointer
LOOP_BY:        INC    DPTR              ;Incrementare pointer
                MOVX A,@DPTR             ;Citeşte primul caracter (număr)
                MOV    SERBUF,A
                PUSH   DPH
                PUSH   DPL               ;Reține în stivă DPTR
                MOV    DPTR,#120H
;Încarcă în DPTR adresa portului pentru afişoarele cu 7 segmente
Biții 3 şi 4 sunt folosiți pentru transferul semnalelor de CLOCK şi respectiv DATE
                CLR    SERPIN1           ;Pune pe zero linia de CLOCK
                MOV    C,BIT_7    ;Transferă în C falg bitul 7 al datei de serializat
                MOV    SERPIN0,C         ;Scrie bitul respectiv ca bit 4 al reg.A
                MOVX @DPTR,A;Scrie la portul de adresă 120H primul bit MSB
                SETB   SERPIN1
                MOVX @DPTR,A             ;Pune linia CLOCK pe 1 logic
;Repetă paşii corepunzători bitului 7 pentru bitul 6 al datei de transmis
                CLR    SERPIN1
                MOV    C,BIT_6
                MOV    SERPIN0,C
                MOVX @DPTR,A
                SETB   SERPIN1
                MOVX @DPTR,A
                CLR    SERPIN1           ;Idem bitul 5
                MOV    C,BIT_5
                MOV    SERPIN0,C
                MOVX @DPTR,A
                SETB   SERPIN1
                MOVX @DPTR,A
                CLR    SERPIN1           ;Idem bitul 4
                MOV    C,BIT_4
                MOV    SERPIN0,C
                MOVX @DPTR,A
                SETB   SERPIN1
                MOVX @DPTR,A
                CLR    SERPIN1           ;Idem bitul 3
```

```
                MOV   C,BIT_3
                MOV   SERPIN0,C
                MOVX  @DPTR,A
                SETB  SERPIN1
                MOVX  @DPTR,A
                CLR   SERPIN1       ;Idem bitul 2
                MOV   C,BIT_2
                MOV   SERPIN0,C
                MOVX  @DPTR,A
                SETB  SERPIN1
                MOVX  @DPTR,A
                CLR   SERPIN1       ;Idem bitul 1
                MOV   C,BIT_1
                MOV   SERPIN0,C
                MOVX  @DPTR,A
                SETB  SERPIN1
                MOVX  @DPTR,A
                CLR   SERPIN1       ;Idem bitul 0
                MOV   C,BIT_0
                MOV   SERPIN0,C
                MOVX  @DPTR,A
                SETB  SERPIN1
                MOVX  @DPTR,A
,0;Reface din stivă pointer-ul de adresare pentru buffer-ul de date
                POP   DPL
                POP   DPH
                INC   NR_OCT        ;Incrementează contorul de transfer octeţi
                MOV   A,NR_OCT
                CJNE  A,#5,LOP_BY ;5 reprezintă numărul de afişoare 7 seg.
;Testează dacă au fost serializate toate informaţiile
                JMP   EXIT          ;Ieşire din rutină
LOP_BY:         JMP   LOOP_BY
EXIT:
```

În legătură cu utilizarea canalului Watch Dog (WD) trebuie să arătăm următoarele: aşa cum s-a precizat în capitolele 1, 2 şi 4 rolul acestui canal temporizator constă în generarea unui RESET-hard atunci când el atinge starea"overflow", respectiv are loc tranziţia de la 11111111B la 00000000B.

Încărcând o anume constantă de timp exprimabilă pe 8 biţi, putem modifica intervalul de timp după care, dacă nu este reîncărcată constanta, respectivul canal temporizator va genera semnalul de RESET.

În cazul nostru, sistemul a prezentat o "sensibilitate" iniţială relativ importantă la zgomotele din mediul industrial. De aceea am adoptat următoarea soluţie: am setat la 256*16μs perioada temporizată de către WatchDog şi periodic în programul de funcţionare, am reanclanşat WD prin reîncărcarea constantei de

timp mai sus precizate. Cum sistemul este un sistem de comandă şi control (comanda elementelor auxiliare ale dozatorului care realizează umplerea automată a sa, precum şi un sistem de măsurare, căci sistemul cântăreşte o anumită cantitate ce este prescrisă pe fluxul tehnologic), cu un număr de 8 stări, acestea au fost numerotate şi stocate împreună cu restul variabilelor atât în memoria internă cât şi în cea extinsă a microcontroller-ului. Actualizarea stării s-a realizat de fiecare dată imediat după ce starea respectivă a apărut. Rutina de inițializare a sistemului a fost modificată astfel încât după fiecare RESET, să analizeze starea sistemului şi funcție de aceasta să realizeze saltul la începutul programului corespunzător stării.

Enumerăm stările principale ale sistemului şi variabilele ce sunt modificate în concordanță cu acestea.

0=stare aşteptare programare / comandă sistem Variabilele: DOZ=0 şi PRG=0
1=stare programare sistem de la tastatură implică Variabilele: PRG=1 şi DOZ=0
2=stare dozare: Variabilele: PRG=0 şi DOZ=1
3=stare temporizare1 Variabilele PRG=0 şi DOZ=0
4=stare temporizare2 Variabilele PRG=0 şi DOZ=0
5=stare temporizare3 Variabilele PRG=0 şi DOZ=0
6=stare aseptare comandă manuală de golire
7=stare temporizare la comanda clapei de golire
8 stare de temporizare pentru blocarea clapetei de golire

Sistemul este supervizat de un calculator de proces ce asigură integrarea sa pe linia tehnologică, dar poate funcționa şi autonom, caz în care operatorul poate interveni pentru a goli manual containerul dozatorului.

Prezentăm în continuare acea parte din programul de funcționare ce realizează directarea corespunzătoare a execuției după ce s-a generat semnalul de RESET.

```
MAIN:
            ANL    IE,#00H              ;Invalidate întreruperile
            MOV    DPTR,#HPRT_O2
            MOV    A,#00H
            MOV    PORT_O2,A
;Opreşte motor ce introduce materie pulverulentă în cântar - comanda este dictată
;de fluxul tehnologic specific
            MOV     R0,#7FH
INIT_1:MOV  @R0,#00H
            DJNZ   R0,INIT_1        ;Inițializare memorie internă cu 0
            ORL    TCON,#0CH
;Validează activarea întreruperilor exeterne de stare 1 pe front - se evită astfel
;reanclanşarea întreruperilor în cazul când nivelul semnalului rămâne prea mult
;timp în 0 logic.
            MOV     SP,#6EH         ;Poziționare stivă sistem la adresa 6EH
;Testare stare sistem - este executată pentru o eventuală resetare ca urmare a
;acțiunii Watch Dog-ului, care generează ceea ce se cheamă "Warm Boot"
```

```
                MOV    R0,#0D0H        ;Locația D0 reține STAREA sistemului[14]
                MOV    A,@R0
;Starea 0 corespunde situației de RESET la pornirea sistemului "COLD RESET"
                CJNE   A,#5AH,PRIM_RESET ;În caz de COLD RESET
;Valoarea 5AH la locația D0H exprimă execuția unui "Warm Boot" ca urmare a
;forțării de către WD a unui semnal de RESET pentru sistem.
                MOV    PORT_O1,#03h
;Reprezintă traiectoria programului în caz de Warm RESET
                MOV    DPTR,#HPRT_O1
                MOV    A,PORT_O1
                MOVX @DPTR,A ;Comandă de refacere stare port "port01" de
        comandă
ERR_RESET:
                MOV     R0,#90H
                MOV     R1,#57H
                MOV     R2,#23
;Pregătire registre pentru transferul pe bloc al informațiilor de stare
;Registrul R0 este pointer-ul sursă, registrul R1 este pointer-ul destinație şi
;registrul R2 este contorul utilizat pentru transferul blocului de date de stare (23D -
;lungimea acestuia), din memoria internă extinsă[15] în memoria SRAM începând de
;la adresa 57H
 ERR_LOOP1:
                MOV     A,@R0          ;Copiază informație sursă în registrul A
                MOV     @R1,A          ;Slavează registrul A în memoria internă
                INC     R1
                INC     R0             ;Incrementează pointerii
                DJNZ    R2,ERR_LOOP1        ;Execută copiere bloc de la 90H
        la 57H
                ORL     PCON,#10H
                MOV     T3,#00H       ;Anclanşare WATCH DOG
;Această operație trebuie realizată periodic la execuția programului pentru a
;preveni generarea unui semnal de RESET
                MOV     A,STARE      ;Locația reține ultima stare postată.
        Indexarea ;stărilor este făcută natural în ordinea de apariție a acestora
;Postarea indexului stării se face întotdeauna după ce starea respectivă a fost atinsă.
```

[14] Am plasat în zona de memorie internă extinsă o copie a tuturor variabilelor de stare importante, căci pe de-o parte memoria RAM internă a fost utilizată până la limita ei fizică, iar pe de altă parte memoria extinsă poate fi cu o probabiliate net mai redusă afectată de erori, căci ea este accesată doar prin instrucțiunea MOV @R0,A (deci indirect)

[15] Procesorul 80C552 prezintă o memorie internă extinsă (256 bytes), dintre care primii 128 sunt identici cu cei ai procesoarelor 8051, iar următorii sunt "umbriți" de zona SFR a UC. Accesarea lor poate fi făcută doar indirect prin intermediul registrului R0.

```
                CJNE   A,#00,ERR_RESET2   ;Testare "COLD RESET"[16] -stare 0
                JMP    ERR_RESET1
ERR_RESET2:
                CJNE   A,#01H,ERR_RESET3 ;Testare - stare 1
                JMP    ERR_RESET1         ;Revenire la starea 1
ERR_RESET3:
                CJNE   A,#02H,ERR_RESET4 ;Testare stare 2 -
                MOV    DPTR,#ERR_END_DOZ       ;Stare DOZARE
                PUSH   DPL
                PUSH   DPH
                SETB   DOZ
;Se reia procesul de dozare cu considerarea greutății dozate până la momentul
;apariției incidentului[17]
                JMP    ERR_DOZ
ERR_RESET4:
                CJNE   A,#03H,ERR_RESET5 ;Testare stare 3
                JMP    ERR_TEMPO1
ERR_RESET5:
                CJNE   A,#04H,ERR_RESET6  ;Testare stare 4
                JMP    ERR_WAIT_STGOL
ERR_RESET6:
                CJNE   A,#05H,ERR_RESET7  ;Testare stare 5
                JMP    ERR_TEMPO3
;Eroare aparută în bucla de aşteptare apăsare buton pentru golirea containerului
ERR_RESET7:
                CJNE   A,#06H,ERR_RESET8  ;Testare stare 6
                JMP    ERR_TEMPO2
ERR_RESET8:
                CJNE   A,#07H,ERR_RESET9   ;Testare stare 7
                JMP    GOLIRE
ERR_RESET9:
                CJNE   A,#08H,ERR_RESET1    ;Testare stare 8
                JMP    ERR_TEMPO_GOL
PRIM_RESET:                         ;Programul de inițializare la Cold Reset
                MOV    R2,#7FH
                MOV    R0,#0FFH
INIT_2:     MOV    @R0,#00H
```

[16] Este necesară testarea redundantă a acestei stări căci apariția semnalelor parazite are caracter aleator

[17] Această stare este tratată deosebit de celelalte, căci este necesară o cântărire distinctă a pulberii aflate în buncărul cântarului dozator, iar pe de altă parte trebuie analizată valoarea curentă a greutății acesteia în raport cu cea prescrisă şi trebuiesc luate acele decizii care se impun, inclusiv generarea semnalelor de eroare în dozare atunci când cantitatea de dozat a fost depăşită.

```
                DEC     R0
                DJNZ    R2,INIT_2
                MOV     R0,#0D0H
                MOV     @R0,#5AH
;ERR_RESET1:
;Inițializare valori prescrise implicite. Reprezintă valorile inițiale care asigură
;starea de bază a sistemului dozator.
                MOV     VAL_PRESCR_L,#40H
                MOV     VAL_PRESCR_H,#1FH
;Urmează instrucțiunile ce detaliază acțiunea sistemului în fiecare dintre posibilele
stări pe care acesta le poate atinge după generarea semnalului de RESET
```

Programul în forma sa completă este memorat pe CD.

În concluzie, putem spune că proiectarea aplicațiilor dedicate impune o deosebită rigurozitate. Nu este suficientă proiectarea corectă din punct de vedere hardware şi software a aplicației, sistemul realizat va trebui să fie integrat corect în mediul, procesul, aplicația mai largă pentru care a fost construit. Complexitatea interacțiunilor ce apar în "viața" sistemului nu poate fi în totalitate simulată, ceea ce face ca etapa de testare "în sistem" o proiectului să aibe o importanță esențială. Nu putem afirma că o anume etapă a proiectării este mai puțin importantă decât alta, dar prin iterarea rațională a etapelor de proiectare, prin testarea cu o colecție cât mai largă de stimuli a sistemului şi prin verificarea sa într-un mediu "zgomotos" putem atinge siguranța necesară funcționari satisfăcătoare a acestuia.

Este necesar, deseori, să construim încă din faza de proiectare odată cu programele de funcționare şi "uneltele" specifice de testare a sistemului. Aceste "unelte" nu sunt altceva decât rutine, programe, uneori complexe, ce asigură validarea corectei funcționări corespunzătoare fiecărei etape pe care sistemul o parcurge în funcționare. Dezvoltarea programelor de testare simultan cu proiectarea sistemului reduce substanțial timpul necesar validării finale.

Nu trebuie neglijat principiul redundanței funcționale -oglindită atât la nivel fizic (hard) cât şi la nivel logic (soft), atunci când dorim să realizăm un sistem robust şi fiabil. Va trebui să verificăm întotdeauna dacă suntem corect "ancorați" în timp, dacă sistemul nostru îşi menține proprietatea de a fi un sistem de reglare "în timp real", asta presupune să verificăm dacă toate informațiile pe care sistemul de comandă le prelucrează sunt "oportune", deci în concordanță cu teorema eşantionării.

Aplicațiile ce sunt detaliate pe CD-ul ataşat lucrării sunt aplicații "reale" ce funcționează în cadrul unor întreprinderi ca: ELCO- SA Tg. Secuiesc, CIMUS SA Câmpulung şi F.S. SA Râşnov. Ele au fost "validate" într-o perioadă de funcționare ce depăşeşte câtiva ani. Aducem mulțumiri celor ce le-au testat şi ne-au ajutat la implementarea acestora, ajutorul lor a fost deosebit de important, iar pentru aceasta le mulțumim